MAGAZINE BSD

FOR NOVICE AND ADVANCED USERS

FREEBSD PROGRAMMING PRIMER

HOW TO CONFIGURE A DEVELOPMENT ENVIRONMENT

THE NEW WAY TO DOWNLOAD *OPERATING SYSTEM CODE AND PORTS*

HOW TO PROGRAM WITH CLOJURE

HOW NEW ZFS UTILITIES *ARE CHANGING FREEBSD & PC-BSD*

HOW TO UPDATE OPENBSD SYSTEM VIA OPENUP

HOW TO SETUP A VIRTUALBOX *VIRTUAL APPLIANCE AS A C++11 DEVELOPER'S MACHINE*

VOL.7 NO.10
ISSUE 10/2013(51)
1898-9144

iXsystems®

855-GREP-4-IX
www.iXsystems.com
Enterprise Servers and Storage for Open Source

✓ Rock-Solid Performance

✓ Professional In-House Support

E5-2600

High Performance,
High Density Servers for
Data Center, Virtualization, & HPC

MODEL: iXR-22X4IB

http://www.iXsystems.com/e5

768GB
of RAM in 1U

KEY FEATURES

iXR-22X4IB	iXR-1204+10G
• Dual Intel® Xeon® Processors E5-2600 Family per node • Intel® C600 series chipset • Four server nodes in 2U of rack space • Up to 256GB main memory per server node • One Mellanox® ConnectX QDR 40Gbp/s Infiniband w/QSFP Connector per node • 12 SAS/SATA drive bays, 3 per node • Hardware RAID via LSI2108 controller • Shared 1620W redundant high-efficiency Platinum level (91%+) power supplies	• Dual Intel® Xeon® Processors E5-2600 Family • Intel® C600 series chipset • Intel® X540 Dual-Port 10 Gigabit Ethernet Controllers • Up to 16 Cores and 32 process threads • Up to 768GB main memory • Four SAS/SATA drive bays • Onboard SATA RAID 0, 1, 5, and 10 • 700W high-efficiency redundant power supply with FC and PMBus (80%+ Gold Certified)

Call iXsystems toll free or visit our website today! **1-855-GREP-4-IX | www.iXsystems.com**

High-Density iXsystems Servers powered by the Intel® Xeon® Processor E5-2600 Family and Intel® C600 series chipset can pack up to 768GB of RAM into 1U of rack space or up to 8 processors - with up to 128 threads - in 2U.

On-board 10 Gigabit Ethernet and Infiniband for Greater Throughput in less Rack Space.

Servers from iXsystems based on the Intel® Xeon® Processor E5-2600 Family feature high-throughput connections on the motherboard, saving critical expansion space. The Intel® C600 Series chipset supports up to 384GB of RAM per processor, allowing performance in a single server to reach new heights. This ensures that you're not paying for more than you need to achieve the performance you want.

The iXR-1204 +10G features dual onboard 10GigE + dual onboard 1GigE network controllers, up to 768GB of RAM and dual Intel® Xeon® Processors E5-2600 Family, freeing up critical expansion card space for application-specific hardware. The uncompromised performance and flexibility of the iXR-1204 +10G makes it suitable for clustering, high-traffic webservers, virtualization, and cloud computing applications - anywhere you need the most resources available.

For even greater performance density, the iXR-22X4IB squeezes four server nodes into two units of rack space, each with dual Intel® Xeon® Processors E5-2600 Family, up to 256GB of RAM, and an on-board Mellanox® ConnectX QDR 40Gbp/s Infiniband w/QSFP Connector. The iXR-22X4IB is perfect for high-powered computing, virtualization, or business intelligence applications that require the computing power of the Intel® Xeon® Processor E5-2600 Family and the high throughput of Infiniband.

HIGH Throughput & INCREDIBLE Performance Density

IXR-1204+10G: **10GbE On-Board**

4 Server Nodes in 2U

IXR-22X4IB

iXsystems®

intel® inside™ Xeon®

Intel, the Intel logo, and Xeon Inside are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.

Call iXsystems toll free or visit our website today! **1-855-GREP-4-IX | www.iXsystems.com**

Dear Readers,

I hope that you are waiting for the October issue and this time you can find a few interesting tips that will help you solve all the technical problems you encounter from time to time.

Today, when you open our magazine, you should read the 9th part of Rob Somerville's series. You will learn what to do and what to add to our CMS to make it more secure and how to refine our interface for adding content. We are also going to gather all the articles in this series and publish a separate issue of BSD magazine devoted to FreeBSD Programming Primer only, especially because many of you read these articles.

For all of you who love the articles written by Michael Shirk, we recommend his publication. Michael teaches you how to download operating system code and ports and how to maintain system source and ports with subversion.

If you need more, you should read the great article: "The revamped Life-Preserver – How new ZFS utilities are changing FreeBSD & PC-BSD" by Kris Moore. In this article, Kris takes a look at a revamped Life-Preserver utility that assists users in ZFS management, including snapshots, replication, mirroring, monitoring and more.

Also in the October issue, we publish a real must read article written by Andrey Vedikhin. Andrey explains how one can setup a VirtualBox virtual appliance as a C++11 developer's machine with the most powerful tools to be available for your C++11 projects.

For the fans of OpenBSD, we have an article on Improved updates and LTS for OpenBSD by Petr Topiarz. Petr describes how to update OpenBSD system via openup and presents what new progressive features there are in OpenBSD.

Last but not least, to encourage all to migrate from Linux to FreeBSD, we recommend reading Charles Rapenne's article. Charles explains the common problems you could have, what you should consider before changing the system and what motivated him for this "adventure". As always, we have a couple of good articles for you. I hope you find them useful and practical.

I would like to thank our Beta Testers and Proofreaders for their excellent work and dedication to help us make this magazine even better. Special thanks to all authors who helped me create every issue. Please keep up the great work and send your articles, tutorials and product reviews, questions, ideas or advises.

Enjoy reading!
Ewa & BSD team

MAGAZINE BSD

Editor in Chief:
Ewa Dudzic
ewa.dudzic@software.com.pl

Contributing:
Michael Shirk, Andrey Vedikhin, Petr Topiarz, Charles Rapenne, Anton Borisev

Top Betatesters & Proofreaders:
Annie Zhang, Denise Ebery, Eric Geissinger, Luca Ferrari, Imad Soltani, Olaoluwa Omokanwaye, Radjis Mahangoe, Mani Kanth, Ben Milman

Special Thanks:
Annie Zhang
Denise Ebery

Art Director:
Ireneusz Pogroszewski

DTP:
Ireneusz Pogroszewski
ireneusz.pogroszewski@software.com.pl

Senior Consultant/Publisher:
Paweł Marciniak
pawel@software.com.pl

CEO:
Ewa Dudzic
ewa.dudzic@software.com.pl

Production Director:
Andrzej Kuca
andrzej.kuca@software.com.pl

Publisher:
Hakin9 Media SK
02-676 Warsaw, Poland
Postepu 17D
Poland
worldwide publishing
editors@bsdmag.org
www.bsdmag.org

Hakin9 Media SK is looking for partners from all over the world. If you are interested in cooperation with us, please contact us via e-mail: editors@bsdmag.org.

FreeBSD Moves to Subversion

FreeBSD users are accustomed to being able to download the entire operating system source code with the ability to compile additional functionality into the kernel and applications. This service had been previously provided by the cvsup and csup programs. Due to the compromise of two FreeBSD cluster servers, the project moved forward with its intention to replace cvsup and csup. This includes the move to subversion.

What you will learn...

- The new way to download operating system code and ports
- Maintain system source and ports with subversion

What you should know...

- Basic FreeBSD knowledge to navigate the command line
- Familiarity with downloading of FreeBSD source with csup/cvsup

The compromise of the FreeBSD servers led to a complete analysis on how developers and users access the software. The issue was caused by an exposed SSH key from a developer that allowed access to one of the legacy build servers (See FreeBSD Compromise of 2012 for reference). The FreeBSD team had to essentially rebuild the binary packages for the FreeBSD 9.1 release because although nothing seemed to have been changed, no backups existed to verify their integrity. With this rebuild, the FreeBSD team moved towards implementing their new process for maintaining source code. At this point in time, the use of `csup/cvsup` is no longer supported.

Listing 1. *Installing subversion using the older package tools*

```
# pkg_add -r subversion
Fetching ftp://ftp.freebsd.org/pub/FreeBSD/ports/amd64/packages-9.2-release/Latest/subversion.tbz... Done.
Fetching ftp://ftp.freebsd.org/pub/FreeBSD/ports/amd64/packages-9.2-release/All/expat-2.1.0.tbz... Done.
Fetching ftp://ftp.freebsd.org/pub/FreeBSD/ports/amd64/packages-9.2-release/All/sqlite3-3.7.17_1.tbz... Done.
Fetching ftp://ftp.freebsd.org/pub/FreeBSD/ports/amd64/packages-9.2-release/All/db42-4.2.52_5.tbz... Done.
Fetching ftp://ftp.freebsd.org/pub/FreeBSD/ports/amd64/packages-9.2-release/All/libiconv-1.14_1.tbz... Done.
Fetching ftp://ftp.freebsd.org/pub/FreeBSD/ports/amd64/packages-9.2-release/All/gettext-0.18.3.tbz... Done.
Fetching ftp://ftp.freebsd.org/pub/FreeBSD/ports/amd64/packages-9.2-release/All/gdbm-1.10.tbz... Done.
Fetching ftp://ftp.freebsd.org/pub/FreeBSD/ports/amd64/packages-9.2-release/All/apr-1.4.8.1.5.2.tbz... Done.
Fetching ftp://ftp.freebsd.org/pub/FreeBSD/ports/amd64/packages-9.2-release/All/serf-1.2.1_1.tbz... Done.
#
```

Listing 2. *Checking out the source code with subversion (In this example, https was used, but the mirrors also support http, svn and rsync). The east coast mirror was used in this example*

```
# cd /usr
# svn co https://svn0.us-east.FreeBSD.org/base/stable/9 /usr/src
Error validating server certificate for 'https://svn0.us-east.freebsd.org:443':
 - The certificate is not issued by a trusted authority. Use the
   fingerprint to validate the certificate manually!
 - The certificate hostname does not match.
Certificate information:
 - Hostname: svnmir.ysv.FreeBSD.org
 - Valid: from Jul 29 22:01:21 2013 GMT until Dec 13 22:01:21 2040 GMT
 - Issuer: clusteradm, FreeBSD.org, (null), CA, US (clusteradm@FreeBSD.org)
 - Fingerprint: 1C:BD:85:95:11:9F:EB:75:A5:4B:C8:A3:FE:08:E4:02:73:06:1E:61
(R)eject, accept (t)emporarily or accept (p)ermanently? p
A    src/sbin/sconfig/sconfig.8
A    src/sbin/clri/clri.8
A    src/sbin/mount_msdosfs/mount_msdosfs.c
.. snip ..
A    secure/lib/libssl/man/SSL_state_string.3
A    secure/lib/libssl/man/SSL_get_client_CA_list.3
A    secure/libexec/ssh-pkcs11-helper/Makefile
A    secure/libexec/Makefile.inc
A    rescue/README
 U   .
Checked out revision 255983.
# cd /usr/src
# svn update
Updating '.':
At revision 255986.
```

Listing 3. *Checking out the ports tree with subversion (Note: the procedure assumes that /usr/ports does not already exist. If /usr/ports exists, "Tree conflicts" will have to be resolved for all of the directories). In this example, https was used with the east coast mirror*

```
# svn co https://svn0.us-east.freebsd.org/ports/branches/RELENG_9_2_0 /usr/ports
   A ports/print/makeindex/distinfo
   A ports/print/latex-tipa/pkg-plist
   A ports/print/latex-tipa/pkg-descr
.. snip ..
   A ports/databases/php55-pdo_pgsql/Makefile
   A ports/databases/py-cmemcache/pkg-descr
   A ports/databases/epgsql/pkg-descr
   A ports/databases/epgsql/pkg-plist
 U   ports
Checked out revision 329041.
# cd /usr/ports
# svn update
Updating '.':
At revision 329043.
#
```

References

- FREEBSD-INSTALL: *http://www.freebsd.org/doc/handbook/bsdinstall.html*
- FreeBSD Compromise of 2012: *http://www.freebsd.org/news/2012-compromise.html*
- Subversion: *http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/svn.html*
- Subversion Primer: *http://www.freebsd.org/doc/en/articles/committers-guide/subversion-primer.html*
- Subversion Mirrors: *http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/svn-mirrors.html*

This article covers the new way to keep up-to-date with the FreeBSD source code. The first thing that needs to be completed is an install of FreeBSD 9.2 (amd64) which was just released (See FREEBSD-INSTALL for installation instructions). Make sure there is at least 6 GB of space available on the hard drive to provide enough space for the system source. This how-to assumes that the system source has not been installed with `bsdinstall`. The first step is to install `subversion`. At the time of this writing, the new package management tool `pkgng` is available, but the older pkg tools will be used to install `subversion`. Listing 1 shows how to install the package for `subversion`.

Users of `cvsup` and `csup` will remember how some configuration was required to pull in the appropriate pieces of code down from various mirrors for a specific release of FreeBSD (examples include src-all, ports-all, etc.). With `subversion`, the code can be checked out from a URL and then maintained by way of the application. `subversion`, allows for additional protocols to be used to check out source code (svn, http, https, rsync). There are currently three mirrors serving up content that provide some redundancy for getting access to code. As time goes on, there should be a large number of mirrors available to use. To maintain the source code, the `svn update` command can simply be run from the path of the system source. Listing 2 shows the initial checkout of the system source code using `subversion`, and an example of updating the stable branch to the latest revision using `svn update`.

With the system source checked out, the building of the base system and kernel is the same as only the software for revision control has been changed. The `Makefile` in the `/usr/src` directory lists the same steps for building the base system and kernel. The example in Listing 2 shows checking out of source code from the stable branch but a specific release can also be checked out. Checking out from `base/stable/9` will download the latest stable code for the 9.x series of FreeBSD, which is currently 9.2. To only follow the 9.1 branch, the following URL may be used: *https://svn0.us-east.freebsd.org/base/releng/9.1/*.

Most FreeBSD users are familiar with the use of `portsnap` to update the ports tree. `subversion` can also be used to keep the ports tree and other branches (such as doc) up-to-date with the latest data. The update steps are the same for the ports tree using `subversion`. Additional port management tools such as `portaudit` and `portmaster` assist with the rebuilding of ports once updated code exists in the tree. Listing 3 shows the similar procedure for checking out the ports tree with `subversion`, and an example of updating the tree to the latest revision using `svn update`.

All of the recommendations in this article are for individuals who wish to build the system or kernel from source. Keeping up-to-date is better done by way of `freebsd-update` for patches to the base OS and `portsnap` for updating the ports tree. `subversion` is just the standard going forward for providing system source to users and developers. The change is subtle and does not add too much overhead to maintaining system source on a FreeBSD server. Though not covered in this article, there are various ways to use the popular developer tool `git` to pull down a clone of the source code, but this is left as an exercise for the reader. These procedures will also work for the upcoming release of FreeBSD 10 which should be released in the first part of 2014.

MICHAEL SHIRK

Michael Shirk is a BSD zealot who has worked with OpenBSD and FreeBSD for over 7 years. He works in the security community and supports Open Source security products that run on BSD operating systems. Michael is the Chief Executive Manager of Daemon Security Inc., a company which provides security solutions utilizing the BSD operating systems: http://www.daemon-security.com.

Professional services and solutions - Imperva, McAfee, HP, Tenable.
Penetration tests, Application Security, Managed Security Services (MSS).

Do as largest companies in Brazil, contact us!

www.ibliss.com.br info@ibliss.com.br +55 11 3255-3926

iBLISS
SEGURANÇA & INTELIGÊNCIA

Gentle Introduction to Programming in Clojure

For those who have never heard of Clojure, let me bring you up to date: Clojure is a Lisp dialect, a language created around 1959 by John McCarthy, being the second oldest programming language next to Fortran.

What you will learn…

- How to program in Clojure.

What you should know…

- Basic level programming knowledge

When Lisp is mentioned, some of you will remember Eric S. Raymond's words on Lisp in his "Homesteading the Noosphere" essay:

"Lisp is worth learning for the profound enlightenment experience you will have when you finally get it; that experience will make you a better programmer for the rest of your days, even if you never actually use Lisp itself a lot."

The clojure.org site (*http://clojure.org/*), defines Clojure in a nut shell.

"Clojure is a dynamic programming language that targets the Java Virtual Machine (and the CLR, and JavaScript). It is designed to be a general-purpose language, combining the approachability and interactive development of a scripting language with an efficient and robust infrastructure for multithreaded programming. Clojure is a compiled language – it compiles directly to JVM bytecode, yet remains completely dynamic. Every feature supported by Clojure is supported at runtime. Clojure provides easy access to the Java frameworks, with optional type hints and type inference, to ensure that calls to Java can avoid reflection. Clojure is a dialect of Lisp, and shares with Lisp the code-as-data philosophy and a powerful macro system. Clojure is predominantly a functional programming language, and features a rich set of immutable, persistent data structures. When mutable state is needed, Clojure offers a software transactional memory system and reactive Agent system that ensures clean, correct, multithreaded designs."

What is so Great About Clojure Then?

Let's take a look:

"Clojure is predominantly a functional programming language" – what does functional programming mean? It's a style of programming where you write all your code using functions that also could take themselves as parameters, and you combine all those functions to perform a single task, that is called code composition. Taking functions as parameters is called high order programming and it makes it easier for code composition to happen. Functions are first-class objects. First, a function is a value (like any other kinds of values). It can be assigned to variables, passed as parameters to functions, be returned as the result of functions, put in collections, and also there are operations that can act on functions. As Lisp has its roots in lambda calculus (programming-like algebra system developed back in the 1930s by Alonzo Church; John McCarthy's work was based on this, you can read his paper *Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I* here: *http://www-formal.stanford.edu/jmc/recursive.pdf*).

A purely functional programming language should obey these mathematical properties of functions:

- Functions have referential transparency, it means that a function always evaluates to the same result as long as it takes the same values as parameters.
- Functions cannot use variables defined outside the function (bye bye globals!)
- Functions do not modify variables or, as is it called in the functional programming, lingo functions do not mutate variables.
- Functions do not do anything that is visible to the outside world (it means no widgets, no screen printing, and so on)
- Functions are closed systems – that means that they do not take information from anything outside them (hard disk, socket, and so on)

If these rules are obeyed then your program is written in a pure functional way. Now, let me tell you about the side effects. In a functional programming paradigm you don't want side effects, they are forbidden, a taboo, you don't speak of them, there are no side effects in a pure functional program, but what is a side effect? A side effect is any piece of code that does something that is visible to the outside world, for example: seeing a YouTube video could be considered as a side effect of a function. This is a *good* side effect or a *bad* one (depends on the video), but Clojure is not a 100% functional language. What fun is there in not seeing your computer doing stuff, like connecting to a database, screen scraping, xml parsing, and so on? It is the side effects of things where you get all the fun. But this is the killer feature that distinguishes it from the rest of the Lisps:

"Clojure provides easy access to the Java frameworks, with optional type hints and type inference, to ensure that calls to Java can avoid reflection."

That means: "anywhere that there is a JVM running, you could stick Clojure to it." It also means that you have access to all the java libraries around to use with Clojure. For me this is among the bestselling points; for example, I spend most of my day on the Z/OS environment (the old mainframe environment, the 72 column hell, the great GUIs we have in there – just kidding, aside I just use Emacs to do all my job there, it is more productive, at least for me), so I have a JVM running on this beast. So, at first chance I had a REPL running on Z/OS and I can leverage all the nice IBM java libraries for interacting with datasets, jcl, and so on. That is really nice.

Enough for an introduction – let's start coding some Clojure. This is what we will need:

- As I'm a member of the church of Emacs, we will use The TRUE editor (Emacs) version 24.
- Emacs Live (*https://github.com/overtone/emacs-live*) – this add-on already has all the Clojure packages you need to be comfortable editing in Clojure, also a lot of other packages preconfigured that are really nice -you could even hack away with Clojure and create music.
- Leiningen (*https://github.com/overtone/emacs-live*) this is the tool for working with Clojure projects, it will download any dependencies needed for your projects, it has many templates for creating new projects. All these goodies are hosted in here: *https://clojars.org/*. You can browse all the libraries available and find the string you need to add to Leiningen so you can start using it.
- Java JDK version 6.

So we have it all setup now. I'll share a simple *.emacs* file. As we are using emacs-live we need to edit or create a `~/.emacs-live.el` file with the following:

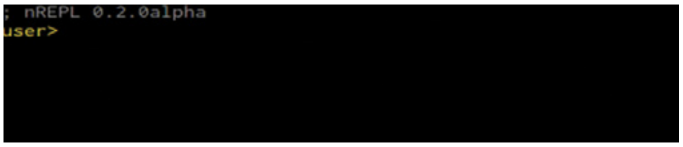
```
;; paredit
 (global-set-key [f7] 'paredit-mode)
 (global-set-key [f9] 'nrepl-jack-in)
```

These are just some keybindings (*http://www.gnu.org/software/emacs/manual/html_node/emacs/Key-Bindings.html*):

F9 will start the REPL, F7 will load the paredit-mode (this is great for editing Lisp, the best thing is when you kill a line, it will not kill the entire line, just the sexp).

Now let's start the REPL. REPL means "read eval, print loop." Here, you can start programming and you will see the results right away.

Press F9 – this will start the repl (Figure 1).

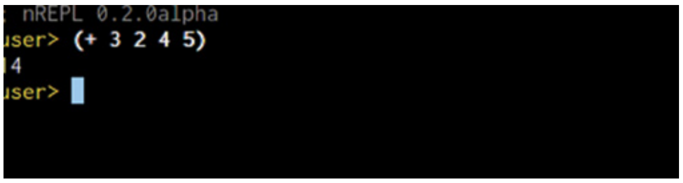


```
; nREPL 0.2.0alpha
user>
```

Figure 1. *Starting REPL*

Clojure, as being a Lisp dialect, has prefix notation – let's type something like a sum (Figure 2):

```
(+ 3 2 4 5 )
```

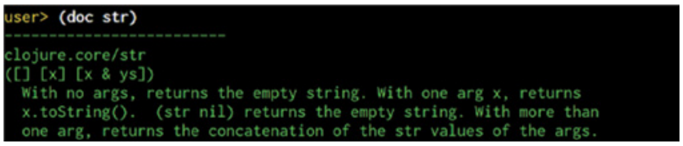


```
; nREPL 0.2.0alpha
user> (+ 3 2 4 5)
14
user> 
```

Figure 2. *Writing a sum in Clojure*

For example if I want to know what a function does, I can just look at the documentation. Just type the following (Figure 3). Let's look at the str function that we use to return strings:

```
(doc str)
```



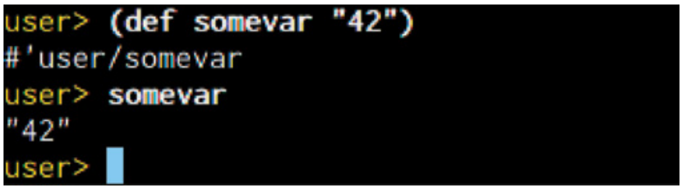
```
user> (doc str)
-------------------------
clojure.core/str
([] [x] [x & ys])
  With no args, returns the empty string. With one arg x, returns
  x.toString().  (str nil) returns the empty string. With more than
  one arg, returns the concatenation of the str values of the args.
```

Figure 3. Analyzing a function

Now, define a variable:

```
(def somevar "42")
```

This variable will live in our namespace. By default, we are in the user namespace *(*`#'user/somevar`*)* (Figure 4).



```
user> (def somevar "42")
#'user/somevar
user> somevar
"42"
user>
```

Figure 4. *Defining the namespace*

Let's look at some data structures.

Strings

Strings are enclosed in double quotes, Clojure strings are java strings.

```
"I'm a string"
```

A Vector

```
[1 2 3]
```

Let's get an element from the vector. As vectors are functions, we could just evaluate the vector to its index.

```
([1 2 3] 0 )
```

You could put anything into them.

```
[1 :two "three"]
```

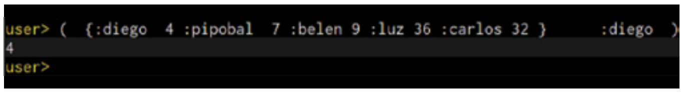
Hashmaps

A hashmap is a map data structure. A map is a set of distinct keys (or indexes) mapped (or associated) to values. The association is such that the map will return the value when given the key. Using a map called a hashmap allows near instant look up time. Here, I created a hashmap where the values represent the ages of people and the keys are the names, for example: Diego is 4 years old.

```
{:diego  4 :pipobal  7 :belen 9 :luz 36 :carlos 32 }
```

As with vectors, maps are functions too so we just evaluate them to their keys. So let's check Diego's age (Figure 5).

```
(  {:diego  4 :pipobal  7 :belen 9 :luz 36 :carlos 32 }
              :diego  )
```



```
user> (  {:diego  4 :pipobal  7 :belen 9 :luz 36 :carlos 32 }      :diego  )
4
user>
```

Figure 5. *Checking Diego's age*

Sets (Unordered and Contain No Duplicates)

```
#{:a :b :c}
```

A List

```
,(1 2 3 x t x s g )
```

Here, we quote the expression. A quote doesn't evaluate this so it returns a list composed of symbols. If you don't put the quote, it will interpret the first element as a function name and pass the rest as parameters.

Vectors, maps, sets, and lists are collections. All Clojure collections are immutable, they are persistent. All collections support count, conj, and seq.

- `count` – returns the number of items in a collection.
- `conj` – add items to the collection; where the item is added depends on the collection type.
- `seq` – returns a seq on the collection; if the collection is empty returns nil.

As all collections support seq, we can use the seq group of functions to go through the entire collection.

The Classic For Loop

```
(for [i (range 7 )] i)
```

Range returns a sequence from 0 to 7 so it is the same if we do this:

```
(for [i [ 0 1 2 3 4 5 6 7 ] i )
```

You can also loop on a condition:

```
(for [ i  (range 10 )  :when (> i 5 )] i )
```

Defining and Calling Functions

You define a function like this:

```
(defn function_name [args]  expresions )
```

If you want variable arguments functions, just do the following.

```
(defn function [ x y & the_rest_of_arguments] expressions )
```

Here the variable *the_rest_of_arguments* will contain a sequence containing all the arguments after the *y* parameter, the *defn* keyword is a macro.

Macros

The word "macro" has a high semantic load. Most people associate a macro with the C/C++ macro concept, which is totally different. The Lisp and Clojure macros are really powerful – it is the same as confusing a wooden slingshot (C/C++ macro) and an AK-47 (Clojure macro). Conrad Barski has had a very good idea in calling them SPEL, which stands for „Semantic Program Enhancement Logic." You can read about his naming convention here: *http://www.lisperati.com/no_macros.html*.

C/C++ macros are just plain text substitution, Clojure macros operate in Lisp data structures. This capability allows Lisp dialects to be homoiconic (this means source code is being expressed as data).

Here we have a C macro:

```
#define seemacroexample(x)  x * (x+5)
```

The C preprocessor will replace any occurrence of MACRO(x) with x * (x+5). Now let's see a Clojure macro or SPEL:

```
(defmacro fn_performance [fname args & body]
     `(defn ~fname ~args
          (time ~@body ) ))
```

First, we define a macro with the *defmacro* keyword – this macro will take as parameters a function name, its arguments, and the rest is the body of the function.

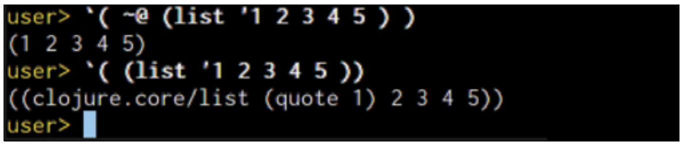
```
`(defn ~fname ...)
```

The backtick (`) – quotes the whole expression, this is called syntax-quote, the expression will not be evaluated.

The Tilda(~) – this is called unquote, this substitutes the value of fname with the value of the argument fname.

Tilda Ampersand (~@) – this means "take the elements of sequence body and splice them in at this point," this unquote splicing is best explained with a little example (Figure 6):

```
`(~@ (list ' 1 2 3 4 5 ))
```



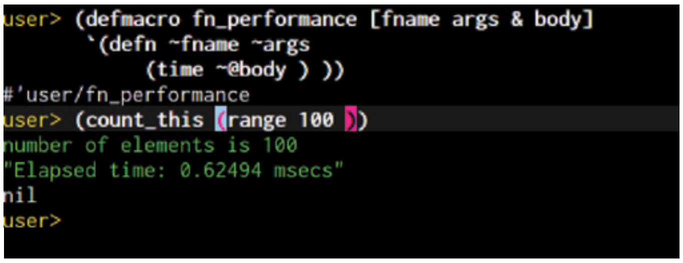
```
user> `( ~@ (list '1 2 3 4 5 ) )
(1 2 3 4 5)
user> `( (list '1 2 3 4 5 ))
((clojure.core/list (quote 1) 2 3 4 5))
user>
```

Figure 6. *Example of unquote splicing*

And finally, the time function evaluates an expression and prints the time it took. It also returns the value of expression. Let's call the macro.

```
(fn_performance  count_this [ x ]
   (println (str "number of elements is " (count x))
       ))
```

After we typed this on the REPL, we have a function in our user namespace called *count_this.* It returns the count of elements in the sequence we pass as parameters, and also prints the time it took to do it.



```
user> (defmacro fn_performance [fname args & body]
       `(defn ~fname ~args
            (time ~@body ) ))
#'user/fn_performance
user> (count_this (range 100 ))
number of elements is 100
"Elapsed time: 0.62494 msecs"
nil
user>
```

Figure 7. *Executing the count_this function*

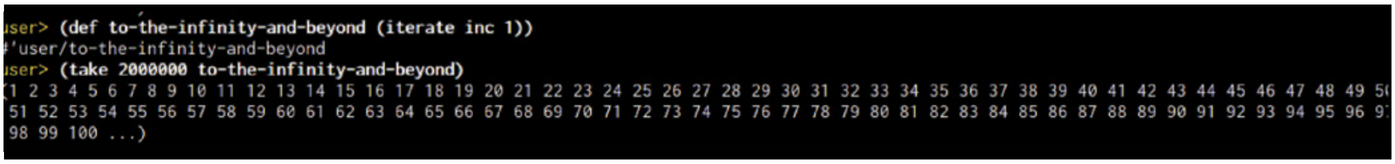
Lazy Evaluation

Also called call-by-need, lazy evaluation means that we delay the evaluation of an expression until it is needed. What are the benefits of that? The benefits of lazy evaluation include:

- Performance increases by avoiding needless calculations, and error conditions in evaluating compound expressions.
- The ability to construct potentially infinite data structures

Infinite data structures, seems possible as we are not creating anything until we are being asked for it, for example:

```
(def to-the-infinity-and-beyond (iterate inc 1))
```



```
user> (def to-the-infinity-and-beyond (iterate inc 1))
#'user/to-the-infinity-and-beyond
user> (take 2000000 to-the-infinity-and-beyond)
(1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 5
 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 9
 98 99 100 ...)
```

Figure 8. *Output of the take function*

We just used the iterate function, according to the Clojure documentation:

```
"Returns a lazy sequence of x, (f x), (f (f x)) etc. f
   must be free of side-effects"
```

So the variable to-the-infinity-and-beyond has numbers from 1 to the infinity?

Let's use the *take* function to take some elements from this sequence, the doc for take is the following (Figure 8):

```
take: "Returns a lazy sequence of the first n items in
   coll, or all items if there are fewer than n."
```

Here is a really subtle difference that makes it all go wrong:

```
"Range: Returns a lazy seq of nums from start (inclusive)
   to end (exclusive), by step, where start defaults to 0,
   step to 1, and end to infinity."
(range  0 2e12 )
```

So we have a lazy sequence from 0 to 2e12, so that means we don't have anything until we call on it, let's just do this:

```
(def  little-seq (range 0 2e12))
(nth little-seq 2000000000)
```

This should fail as we created a reference on *little-seq* that points to the first element to the sequence, so the JVM garbage collector will not free up the elements in *little-seq* as it is being walked through to get to the *nth* element.

Instead as in the (`nth  (range  0  2e12)  20000000000`), here the JVM garbage collector can free the elements as we don't have any reference to it, so the JVM is free to free those up.

Summing it Up

This is the end of this introduction. There is a lot that I have not covered, but for a gentle introduction, I hope this sets you in the right state of mind to program with Clojure and gets you interested in the language. You can do web developing in Clojure, there is a statistics framework also, data mining through the weka framework, there is a lot of cool stuff to do in clojure and the language is not going anywhere nor is vaporware. Now it has enterprise support with *http://cognitect.com/* as they just hired the creator of Clojure – Rich Hickey. Many great tutorials and information are in the following URLs:

- *http://en.wikibooks.org/wiki/Clojure_Programming/Examples/*
- *http://clojuredocs.org/*
- *http://clojure-doc.org/articles/tutorials/introduction.html*

You could also take a look at the book entitled *Land of Lisp.* It is a great book written by Conrad Barski, it is not Clojure based but it will give you some nice introduction to Lisp languages.

CARLOS ANTONIO NEIRA BUSTOS

Carlos Antonio Neira Bustos is a C, Unix and Mainframe developer. He develops in asm and does some kernel development for a living. In his free time he contributes to open source projects. Apart form that, he spends his time on testing and experimenting with his machines. What gives him a great fun is solving the old problems with new ideas. You may reach him at: cneirabustos@gmail.com.

Is your
MISSION-CRITICAL
security strong enough
to stop a
SKILLED ATTACKER?

Don't guess
Don't believe
Don't hope

KNOW!

acros

An ACROS Penetration Test is conducted exactly like a real attack by a skilled, motivated adversary – only without the damage. We will find the weakest links in your security and use all our knowledge, skills and capabilities to try to achieve exactly what your security measures and policies are there to prevent. If it sounds difficult, we're interested.

Experience the ultimate test of your security.
(After all, the only alternative is to wait for an actual attack.)

ACROS Security – http://www.acrossecurity.com – security@acrossecurity.com

The Revamped Life-Preserver

How New ZFS Utilities are Changing FreeBSD & PC-BSD

Since the inclusion of ZFS into the FreeBSD base system, it has revolutionized how enterprise users have managed their data. However due to higher memory requirements and the difficulty of the initial setup, it was often out of reach for less experienced system administrators and more modest system hardware. However, over the past several years ZFS on BSD has greatly matured, reducing the complexity of the initial setup and tuning required to perform optimally. In early 2013, this led the PC-BSD project to re-focus and fully embrace ZFS as its default and only file-system for both desktop and server deployments. This decision immediately spawned development of a new class of tools and utilities which assist users in unlocking the vast potential that ZFS brings to their system, in areas of data-integrity, instant backup and restore, fail-over, performance and more. In this article, we will take a look at the revamped Life-Preserver utility which assists users in ZFS management, including snapshots, replication, mirroring, monitoring and more.

The Initial ZFS Setup

Starting in PC-BSD 9.2 and later, any installation performed with its installer will automatically set up ZFS as the default root file-system. This includes installations of both a Desktop (PC-BSD) or Server (TrueOS). The default layout will be created with a SWAP partition, (mainly for crash-dumps) and the rest of the disk / partition allocated to a single ZFS zpool. During the install, options will be provided for the setup of multiple disk drives through ZFS mirroring or raidz levels 1-3. In addition the GRUB boot-manager will be installed, which will facilitate the usage of ZFS Boot-Environments.

Life-Preserver

Starting in PC-BSD 9.2 and later, a new “lpreserver” (Life-Preserver) CLI utility has been included. For FreeBSD users this utility can be installed from the ports tree in *system/pcbsd-utils*. This utility provides a framework to easily manage other aspects of ZFS, such as its abilities to create snapshots, replicate data, mirror drives and more. We will first take a look at snapshot creation and scheduling.

Snapshots

Due to ZFS being a “copy-on-write” file-system, one huge benefit is the ability to instantly create “snapshots” of your data. A snapshot is simply a “meta-marker” of how your data appeared at any given moment in time. When a snapshot is created, ZFS will ensure that those portions of the disk which existed at the time of the snapshot will not be overwritten until the snapshot has been destroyed. Thus initially the snapshot will not consume any space on the disk, and grow only as the current data changes from the original snapshot state. Using the “lpreserver” utility we can easily setup a schedule of creating / pruning snapshots.

```
# lpreserver cronsnap tank1 start daily@22 10
```

The above command will use cron to schedule a set of daily snapshots to be created at 22:00 on the "tank1" zpool. The last number "10" is used to indicate the total number of snapshots to keep. When this number is exceeded, the oldest snapshot will be pruned automatically, reclaiming any disk space which may have been allocated to that snapshot. By default all snapshots created will be recursive, meaning that any additional ZFS datasets created on the "tank1" zpool will also have snapshots created / pruned automatically. While the recursive option can be disabled, most users will not want to do this, in order to keep the system intact when doing replication.

Replication

With our snapshots being created on a regular basis we in essence now have access to an instant on-disk backup. But what happens if the system disk dies, or we lose the entire computer due to theft or some natural catastrophe? Luckily ZFS also has the ability to replicate datasets and snapshots across the wire to a remote location. Using the "lpreserver" command it is easy to setup a replication target for your zpool.

```
# lpreserver replicate add backupsys backupuser 22
              tank1 zpoolbackup/backups sync
```

The above command usage is fairly simple. In this example we have scheduled a replication to the "backupsys" host, connecting with "backupuser" via SSH on port 22. Our local "tank1" and its dependents will be replicated to the remote dataset "zpoolbackup/backups" and it will be kept in "sync" with local snapshot creation. Before this replication can occur a few things must take place on the remote side. First, the remote system must also have a valid ZFS zpool, which is compatible with the version of ZFS on your local machine. Second, you must ensure that the user you are connecting with has permissions to create the ZFS datasets on the remote side:

```
# zfs allow -u <user> create,receive,mount,userprop,des
              troy,send,hold <remotedataset>
```

Lastly we will need to set up authorized SSH keys between the local and remote system, allowing passwordless logins to take place. With these steps followed the local system will now have access to "replicate" the data from your local machine to the target remote dataset. This ensures that even if something should happen to the local box, a copy of your data will still be retrievable from a different location. Replications are sent incrementally, and can be very quick, but the initial sync may take some time depending upon disk speed and network conditions. During the replication phase automatic snapshot pruning will be halted in order to prevent removing the replicating snapshots. Next we will take a look at how you can easily restore or clone a system from a remote replication.

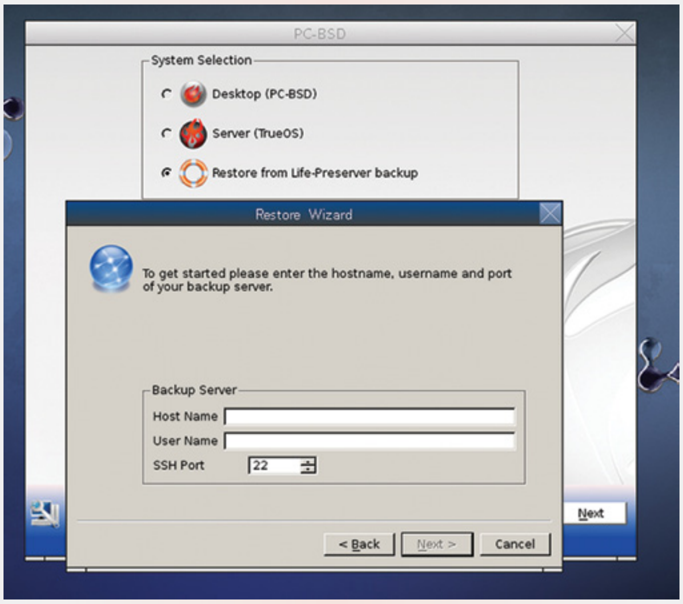
Pro Tip

Currently with ZFS if a replication fails you may need to re-initialize the remote side to avoid it getting "stuck". To do so, use the command: "lpreserver replicate init `<localdataset>`". In an upcoming version of OpenZFS it will support resuming the send/receive command, making this less necessary.

Restoring

Booting a PC-BSD 9.2 or later install media gives you the option to not only install fresh, but restore from a Life-Preserver replication.

During the installation you may select the "Restore from Life-Preserver backup" option to begin the process. The graphical wizard will then walk you through the process of connecting to the backup server, as well as setting options for restoration. A helpful feature is the ability to change your initial ZFS pool options during a restore. This can be used in the case of upgrading to a larger disk, or migrating from a single-disk system to a ZFS software raid, using all the replicated data you previously saved. Once the restore is finished, you will be able to reboot the system back to the state it was in during the last replication.



To get started please enter the hostname, username and port of your backup server.

SSH Port 22

Figure 1. *Restoring from a Life-Preserver backup*

Mirroring

In addition to being able to snapshot, replicate and restore your data, you may need a more "high-availability" solution. During the initial system installation you are given options to create a software raid, either in a direct mirror configuration, or using "raidz" levels 1-3. However if you only installed to a single disk, you can still take advantage of ZFS mirroring at any time. To get started you will need a second disk drive, the same size as the target disk or larger. This second disk can be connected internally or externally via USB 3.0, eSATA or any other method which presents a raw disk device to your system. With the disk drive connected, it is now possible to "attach" it to your existing ZFS zpool using the following command:

```
#  lpreserver zpool attach tank1 /dev/da0
```

This command will take the disk drive `/dev/da0` and attach it to the "tank1" ZFS zpool. During the setup, this disk will be wiped and new partitions created to match the first disk in your existing zpool configuration. Once the drive is attached, it will begin to sync up with the existing zpool data, known as the "resilvering" phase. This may take some time depending upon the speed and size of your disks. Additionally the disk will be stamped with GRUB, making it bootable. The advantage of this is that should your original disk in the array fail, you can continue to run from the second drive without any data loss. If you need to shut the system down and swap-out the bad disk, booting is still possible from the second drive. The replaced disk drive can then be attached in a similar manner, which will re-sync data from the secondary disk. This method can be used to attach as many disks as you require, giving you any level of extra redundancy that your system may need. As long as a single disk in the array remains intact your data will be safe and the system bootable.

In the case of using external disks, it is even possible to disconnect and re-connect at a later time using the `lpreserver` command:

```
# lpreserver zpool offline tank1 /dev/da0s1a
# lpreserver zpool online tank1 /dev/da0s1a
```

When the disk is re-connected and set in the "online" mode, the resilvering process will begin again, re-syncing the external disk incrementally. Using the "offline/online" method is recommended when you plan on re-attaching the disk, otherwise you may need to resilver all the data over again. This can be used in the case of a portable machine, such as a laptop, which is connected to an external disk drive periodically for backups.

References

- PC-BSD pkgng repo – *http://wiki.pcbsd.org/index.php/Turn_FreeBSD_into_PC-BSD%C2%AE*
- PC-BSD GitHub – *https://github.com/pcbsd/*
- PC-BSD Mailing Lists, Forums and Bug-Tracker – *http://lists.pcbsd.org, http://forums.pcbsd.org/, http://trac.pcbsd.org/*

Monitoring

Last but not least, the "lpreserver" utility can also provide basic reporting about snapshots and your disk status. By using the `lpreserver set email <address>` command it is possible to enable reporting about the status of your system. Life-Preserver can be setup to send either all messages, warnings, or errors only using the `lpreserver set emailopts ALL/WARN/ERRORS` command. Currently Life-Preserver can report the results of automatic snapshots and replications, as well as provide warnings when a disk has failed, or if the zpool is running low on disk space. If your network only allows the usage of a smtp server, you can install the `ssmtp` utility to enable mailing through that gateway.

Conclusion

In this article we have taken a look at how the Life-Preserver utility can leverage ZFS to assist in snapshot creation, replication and restoring. By using these tools, users can better manage and plan for disasters on desktops, laptops or servers in a variety of unique situations. While the Life-Preserver utility is included in all installs of PC-BSD or TrueOS, it is also available to FreeBSD users in the public PC-BSD pkgng repo or via ports in `sysutils/pcbsd-utils`.

KRIS MOORE

Kris Moore is the founder and lead developer of the PC-BSD project. He is also the co-host of the weekly BSDNow! video podcast with Allan Jude of ScaleEngine. When not at home programming, he travels around the world giving talks and tutorials on various BSD related topics at Linux and BSD conferences alike. He currently lives in Tennesee (USA) with his wife and five children and enjoys video gaming in his (very limited) spare time.

UPDATE
NOW WITH
STIG
AUDITING

"IN SOME CASES **nipper studio** HAS VIRTUALLY **REMOVED** the **NEED FOR** a **MANUAL AUDIT**"

CISCO SYSTEMS INC.

Titania's award winning Nipper Studio configuration auditing tool is helping security consultants and end-user organizations worldwide improve their network security. Its reports are more detailed than those typically produced by scanners, enabling you to maintain a higher level of vulnerability analysis in the intervals between penetration tests.

Now used in over 45 countries, Nipper Studio provides a thorough, fast & cost effective way to securely audit over 100 different types of network device. The NSA, FBI, DoD & U.S. Treasury already use it, so why not try it for free at www.titania.com

2012 Computing Security Awards
WINNER
Enterprise Security Solution of the Year

Security Products' GOVIES
2013 Government Security Awards

2012 Computing Security Awards
WINNER
Network Security Solution of the Year

www.titania.com

Migrating from Linux to FreeBSD

In the following article, I will share with you what I learned when I have converted my servers from Debian to FreeBSD. I will explain the common problems you could have, what you should consider before changing the system and what motivated me for this "adventure".

What you will learn…

- Compatibility problems between Debian and FreeBSD.

What you should know…

- System administration basis
- Some development knowledge.

To begin this article, I need to set up the background, so that you can fully understand what follows. I am a system administrator in a small company. I work every day with two developers using *my servers.* Their knowledge about operating systems is quite good. I had to renew a whole set of servers, used for a high availability service for our biggest client.

So far, everything is running with Debian GNU/Linux, but I am always talking about FreeBSD to my colleagues. This time, I proposed switching to FreeBSD for our new servers!

What follows is about decisions, choices and "philosophy". This article is not a how-to switch from Debian to FreeBSD. It may help you decide what to use; it may help you find answers; but you won't find any commands or scripts.

I am trying to write this article while being as neutral as possible. I may make some mistakes in my choices and reasons, but they are mine. You can send me an email if you want to share a thought or if you think that I said something wrong.

Why change?

I knew that I would need a good reason to switch operating systems, because this will certainly create extra work with regard to compatibility, finding packages, and learning the operations of the operating system.

That is why I made a Pros/Cons list to explain why I would like to use FreeBSD for the next few years. I tried to be as honest as possible, not trying to sway you to FreeBSD with only pros, while Linux would have had only cons. Linux has been working well, so why change it?

Pros and Cons list:

- FreeBSD and Linux are equivalents and they basically perform the same. As we are using open-source software only, we should not encounter compatibility problems, but we must be aware that if we need to use proprietary tools, it may not work.
- Switching to FreeBSD may not be that easy. We may have to change some of our source code and may need to teach people new operations or features. Some user commands may be different.
- FreeBSD is a Unix system, keeping Unix philosophy. System administration does not change with every new release. Linux, over the last few years, has been evolving and moving away from the Unix philosophy. Every new release changes many components, or includes more services and applications. I have the feeling that things may become more complicated in the future.
- FreeBSD is able to provide high-availability service, with two lines of config per host. We do not have to struggle with heartbeat/pacemaker/corosync.

- FreeBSD is using a port tree. Compilation takes time, but it will be centralized with the software Poudriere. We will be able to choose compilation flags, the versions of postgresql and PHP that we want to use. We have had surprises with Debian in the past: old PostgreSQL, PHP missing some features.
- FreeBSD is developed as a complete package and is coherent, with man pages that are up-to-date. The FreeBSD handbook is a good source of documentation and best practices also.

As you see, my list is mostly talking about philosophy concerns rather than performance or technical differences. In my company, we like to keep things simple, and we have the feeling that Linux is going in the opposite direction.

The deal

My boss accepted the idea that we move to FreeBSD. We agreed that this should work with as few modifications as possible of scripts / software / makefiles / users habits, etc. For example, For example, I do not have time to fix every Bash script, but adding a symlink could fix the problem -> hello /bin/bash! I really care about my users (the developers); if they want Bash, I have no problem with it and I will provide them with Bash with the same alias as before.

The migration: from Debian to FreeBSD

In this part, I shall share some issues that I had, and how I fixed them.

The Shell

An important part of the system is the shell. By default, FreeBSD uses the unfriendly csh, which does not keep history and no color. I will not be able to advocate FreeBSD to my users with just csh. I decided to install bash as we are used to and importing our bash config files.

Where are my commands?

You would think that people would understand that if you make a change to the operating system, then they will need to learn about this, but they don't. A developer asked me "The command *free* doesn't work, why?"

Under Linux, this command shows you information about memory usage. I just told him: "*free* does not exist in FreeBSD. You can use the vmstat command and you will get everything you need. You can also use top." Users will have to learn a few new tools.

What's wrong with putty?

On Windows, you are likely to use the ssh client *Putty*. I have to admit that Putty with the default FreeBSD profile is not user friendly at all. A lot of keys are not doing what you would expect them to do. For example, typing "home" or "end" will display a "~", typing "delete" will add a character, console dialogs (like the one in ports or the frame in tmux) are made of weird characters etc...

After a quick search on the Internet, the fix was made on the server-side. People are now happy with their Putty.

Paths of death

One extremely important change between the systems is the filesystem hierarchy. In FreeBSD, configurations files are separated between `/etc` and `/usr/local/etc`, while Debian stores everything in `/etc`. The folder `/var/www` does not exist, it is replaced by `/usr/local/www`. Packages binaries are under `/usr/local/bin/`.

Everything installed on the system with packages is installed in `/usr/local/`. Files outside this folder come from the base system.

B.1 Compilers

Our work uses compilers and we rely on C and C++. We had a problem with a lot of makefiles, using the compiler "gcc". "gcc" command was a 4.6 GCC on the previous system, while on FreeBSD "gcc" is a 4.2.1 version. If we add a new compiler, its full name must be used e.g.: gcc46 for GCC 4.6 etc.

I had to fix manually some scripts where "gcc" was not compiling, due to compatibility problems.

In addition to the outdated GCC, I have been struggling with include and libraries paths too. The linker could not find the libraries located in `/usr/local/lib` instead of `/usr/lib` as before. The same kind of error was happening with includes files.

After some hours, I decided to use an environment variable telling GCC to add a path by default to fix the problem. I know this solution makes the makefile not very portable because the paths will not be right without the environment variable. But, the makefile was not actually portable, and we do not plan to share our code.

We have scripts!

We have a lot of scripts, doing various things. Every script is using `#!/bin/bash`!

You can imagine how boring it would be to fix every script manually. That is why someone wrote a "sed" script.

I preferred to link Bash to `/bin/bash`, which is the path that our previous scripts were expecting. People did not have to waste time searching why their scripts using `/bin/bash` do not work.

Please, when you write scripts, use `#!/usr/bin/env bash` instead of bash full path. And if you use Linux, if you link to `/bin/sh`, be sure to really be using `/bin/sh` and not a disguised Bash.

FreeBSD crash course

When the time comes to change users tools, at least, be kind. Explain to them what changed and why you did it and teach them the basis of their new tool. Like I said before, the command "free" does not exist, binaries are in `/usr/local/bin/` path, services scripts are either in `/etc/rc.d` or `/usr/local/etc/rc.d`, `/etc/rc.conf` is the main configuration file etc. My users were happy to learn a few things about their new FreeBSD system, they did not feel lost or abandoned. Take time to explain to them what changed, give them some time to learn and it is all good.

Where are my packages

I had some fun (or bad news, I do not know) with the packages. We are *mostly* using open-source software, or our own programs.

Hello, I am an old closed source Linux library

I found *ONE* program relying on a 1997 closed source Linux library that we *MUST* use. Even on the previous Debian server, we had to do some hacks to make it work! This library is very important to our business and we *must* use it. If we cannot use it, bye bye FreeBSD.

Luckily, I got it working with Linux emulation and some dirty hacks that I would have preferred to skip. Also, in a few months, we will trash it because it is getting replaced.

This is a bit frustrating to spend a lot of time getting a program working, when you know it will be removed soon.

Hello, I am away from ports

The other case is funny too. I was installing the dependencies of our application and I knew exactly what was needed, installing them one by one.

…Until that moment when I could not find the latest library in ports! I have been looking on my favorite search engine "mylibrary + freebsd", and found no answer.

I have been cursing the developers. "Where did you find this? No one is using it! It's not even in the freebsd ports!" And then I had to learn how to write a port that compiled. (I know I should share my port, but it is an ugly one actually).

Surprises!

Now, all my dependencies are installed, services are running awaiting to be checked. (Yes, you should really consider checking if everything works as expected when you change your operating system and your servers!

Then, when everything is done, someone comes to tell you he found something strange.

My program A does not work as expected

We have a program that stopped terminating correctly on FreeBSD. This seems to come from a difference with threads, and exiting was not killing all the threads. We made a little change so the application was terminating correctly with Ctrl-C.

My program B consumes 30% less memory

I do not have an explanation for this one. One of our applications consumes a lot of memory by loading shape files (with libshape), and with the same data, the new FreeBSD consumes 30% less memory than the previous Debian system.

I assume this difference is coming from the shapelib, which is newer on the FreeBSD system, and which is compiled with optimizations for the server itself. We were very happy to see this behavior!

Valgrind needs to be recompiled…

To debug the application, I was telling before, we used valgrind. After a day spent running the program under valgrind, we were not able to use the results – valgrind gave us an error. Valgrind has been monitoring too much memory that a compile-time limitation was reached. We had to recompile Valgrind with a modification in the source code to increase a value.

I have been happy to tell the developers that ten minutes later, I compiled valgrind from ports with a patch of mine, and that it was installed on the servers!

Conclusion

In a nutshell, I would say that FreeBSD gives us more freedom than our previous systems. I can really provide the developers with the tools they want.

But, when it comes to using proprietary tools, I have to admit that FreeBSD may not be the best option. I have not had this problem (except for one library).

If I would have been using Solaris, NetBSD, or another Linux distribution, I think I could have had a working result. But FreeBSD comes with good performance, server hardware compatibility, a complete handbook, incredible stability, control over packages and a clean and well-designed system.

CHARLES RAPENNE

*Charles Rapenne is a Linux, and now FreeBSD system administrator in a small company in France. He enjoys using *BSD systems and writing applications in Common LISP. His servers rely only on open source software.*

VERISIGN®

vBSDCon · SAVE THE DATE!

DULLES, VA · OCTOBER 25-27, 2013

Please join us October 25-27, 2013 at the Hyatt in Dulles, Virginia for the first biennial vBSDCon event. This exciting weekend will bring together members of the BSD community for a series of roundtable discussions, educational sessions, best practice conversations, and exclusive networking opportunities. See below for details on this industry weekend not to be missed:

AGENDA

- **Friday, October 25:** Evening Reception
- **Saturday, October 26:** General Session, Birds of a Feather Sessions
- **Sunday, October 27:** General Session, Breakout Sessions

WHO SHOULD ATTEND

- Developers
- Engineers
- Administrators
- Innovators

TOPICS

- PkgNG w/ Baptiste Daroussin
- A comprehensive look at bsdinstall with Devin Teske
- Netflix Demo/Presentation with Scott Long
- netmap with Luigi Rizzo
- Migration from GCC to LLVM/Clang with David Chisnall

REGISTRATION INFORMATION WILL BE SENT TO YOU IN MAY!

Questions? Please contact: eventsteam@verisign.com

VerisignInc.com

FreeBSD for C++11 Developer (Eclipse Indigo + CDT + GCC 4.8)

FreeBSD is a very stable, high performing and extremely secure operating system. It is not a surprise that a lot of developers create applications for this platform. However, a Desktop Manager and IDE for C++ projects with other development tools (profilers etc.) are not installed in the default setup. If you are not running FreeBSD 10, then even the compiler in base is too old (GCC 4.2) and it does not support the new C++11 standard. You will also not be able to efficiently use the GDB debugger from the base because its version is also too old and it does not understand new C++11 features as well.

What you will learn…

- How to update and secure your system.
- How to install x11/gnome2 as your desktop manager.
- How to install GCC 4.8 (which fully supports the new C++11 standard) and the latest GDB debugger.
- How to install IDE Eclipse Indigo + CDT.
- How to configure your system and Eclipse to support C++11 applications.
- How to enable shared folders in VirtualBox.
- How to install and use devel/valgrind tools.
- How to install and use gprof profiler.

What you should know…

- How to install FreeBSD.
- FreeBSD basics.

This article explains how one can setup a VirtualBox virtual appliance as a C++11 developer's machine with the most powerful tools to be available for your C++11 projects.

Although my main focus will be on C++11, this article still might be useful for those who would like to use Eclipse IDE for Java, Python, PHP etc. programming.

Step 1. Prepare your system

FreeBSD 9.1 RELEASE installation

Install FreeBSD from CD/DVD but do not forget to create your own user and add this user to the `wheel` group.

Install ports tree

```
# portsnap fetch extract
```

In the future you will be able to update your ports tree with the command below:

```
# portsnap fetch update
```

Install Subversion

To install and update the FreeBSD kernel/world sources we will need to use `devel/subversion`. Let's install it. First of all, install `ports-mgmt/pkg` and `ports-mgmt/portmaster`:

```
# cd /usr/ports/ports-mgmt/pkg
# make install clean
# cp /usr/local/etc/pkg.conf.sample /usr/local/etc/pkg.conf
# pkg update
```

You can safely ignore any warnings.

```
# cd /usr/ports/ports-mgmt/portmaster
# make install clean
```

Accept default options when asked.

```
# echo 'WITH_PKGNG=yes' >> /etc/make.conf
# pkg2ng
```

Then install Perl 5.16:

```
# cd /usr/ports/lang/perl5.16
# make config-recursive
```

Select the "Threads" option. Leave the rest of the configuration options at their default values.

```
# make install clean
```

Then install `devel/subversion`:

```
# cd /usr/ports/devel/subversion
# make config-recursive
```

Make sure that the "Serf" option is chosen in the Subversion options. This option enables connections over http/https. Leave the rest of the configuration options at their default values.

```
# make install clean
```

Install FreeBSD kernel/world sources

```
# rm -R /usr/src
# mkdir /usr/src
# cd /usr/src
# svn co http://svn.freebsd.org/base/releng/9.1/ /usr/src/
```

I assumed that you have installed FreeBSD 9.1. If not, change the version accordingly.

It will take a while...

In the future you will be able to update your FreeBSD sources with the following command:

```
# cd /usr/src
# svn up
```

Upgrade your FreeBSD kernel/world (apply security patches)

```
# freebsd-update fetch
# freebsd-update install
```

It may give you additional instructions on what to do. Follow them. At the end do not forget to reboot:

```
# shutdown -r now
# uname -a
```

Make sure that you are running the latest version with all the security patches applied. You will need to repeat this step in the future when new security holes are discovered in FreeBSD kernel/world. Alternatively, you can upgrade your system by compiling and installing a custom (or generic) kernel as described below.

- Clean up directories:

```
# cd /usr/src
# make cleanworld && make cleandir
```

- Build world and kernel:

```
# make buildworld
# make buildkernel KERNCONF=YOUR_KERNEL_CONFIG_NAME
```

- Install the new kernel:

```
# make installkernel KERNCONF=YOUR_KERNEL_CONFIG_NAME
# shutdown -r now
```

- Boot into single user mode and install new world:

```
# adjkerntz -i
# mount -a -t ufs
# mergemaster -p -U
# cd /usr/src
# make installworld
# make delete-old
# mergemaster -U
# shutdown -r now
```

- If you are updating between major releases, you should rebuild all your ports now or at least install the proper `compat` libs in the meantime.
- Delete all libraries (in case you have rebuilt ports after a major release update):

```
# cd /usr/src
# make delete-old-libs
```

Install Sudo

```
# cd /usr/ports/security/sudo
# make config-recursive
```

Leave the configuration options at their default values.

```
# make install clean
```

Uncomment the following line in the `/usr/local/etc/sudoers`:

```
%wheel ALL=(ALL) ALL
```

Make sure `sudo` works:

```
# su your_user_name
% sudo sh
# exit
% exit
```

In order to disable the root password, find the root entry in the `/etc/master.passwd` file and replace it with the line below:

```
root:*:0:0::0:0:Charlie &:/root:/bin/csh
```

Update the internal passwords database:

```
# pwd_mkdb -C /etc/master.passwd
# pwd_mkdb -p /etc/master.passwd
# pwd_mkdb /etc/master.passwd
```

Install OpenNTP

OpenNTPD provides the ability to sync the local clock to remote NTP servers and can even act as an NTP server itself, redistributing the local clock.

Configure `ntpdate` to set the clock at boot time:

```
# echo 'ntpdate_enable="YES"' >> /etc/rc.conf
# echo 'ntpdate_hosts="pool.ntp.org"' >> /etc/rc.conf
```

Install `net/openntpd`:

```
# cd /usr/ports/net/openntpd
# make config-recursive
# make install clean
```

The default configuration will suffice for our purposes but feel free to edit `/usr/local/etc/ntpd.conf` if needed, in particular to change ntp server.

Configure `openntpd` to start at boot time and start it now:

```
# echo 'openntpd_enable="YES"' >> /etc/rc.conf
# /usr/local/etc/rc.d/openntpd start
```

Step 2. Install Gnome 2

Install Xorg

```
# cd /usr/ports/x11/xorg
# make config-recursive
```

Leave the options at their defaults.

```
# make install clean
# echo 'hald_enable="YES"' >> /etc/rc.conf
# echo 'dbus_enable="YES"' >> /etc/rc.conf
# Xorg -configure
```

Edit the "Screen" session of `/root/xorg.conf.new` as below:

```
Section "Screen"
    Identifier "Screen0"
    Device "Card0"
    Monitor "Monitor0"
    DefaultDepth 24
    SubSection "Display"
        Viewport 0 0
        Depth 24
        Modes "1024x768"
    EndSubSection
EndSection
```

Change the screen resolution (in red) to the one supported by your display and then copy your new configuration file to the right location:

```
# cp /root/xorg.conf.new /etc/X11/xorg.conf
```

Install Gnome

```
# cd /usr/ports/x11/gnome2
# make config-recursive
```

Leave the options at their defaults.

```
# make install clean
# echo 'proc /proc procfs rw 0 0' >> /etc/fstab
# echo 'gdm_enable="YES"' >> /etc/rc.conf
# reboot
```

Install VirtualBox Guest Additions

First, install the `emulators/virtualbox-ose-additions` port:

```
# cd /usr/ports/emulators/virtualbox-ose-additions && make
                install clean
```

Add these lines to `/etc/rc.conf`:

```
vboxguest_enable="YES"
vboxservice_enable="YES"
vboxservice_flags="--disable-timesync"
```

Re-configure Xorg:

```
# cp /etc/X11/xorg.conf /etc/X11/xorg.conf.old
# /usr/local/etc/rc.d/gdm stop
# Xorg -configure
# cp /root/xorg.conf.new /etc/X11/xorg.conf
```

Make sure that VirtualBox devices are used for mouse and video. Follow the instructions on *http://www.freebsd.org/doc/en/books/handbook/virtualization-guest.html* (the "22.2.4. VirtualBox™ Guest Additions on a FreeBSD Guest" section). Edit the "Screen" session of `/etc/X11/xorg.conf` as below:

```
Section "Screen"
    Identifier "Screen0"
    Device "Card0"
    Monitor "Monitor0"
    DefaultDepth 24
    SubSection "Display"
        Viewport 0 0
        Depth 24
        Modes "1024x768"
    EndSubSection
EndSection
```

Start Gnome:

```
# /usr/local/etc/rc.d/gdm start
```

Step 3. Install C++11 compliant compiler and debugger

We can either use GCC 4.8 (or later) or Clang.
  Let's use GCC:

```
# cd /usr/ports/lang/gcc48
# make config-recursive
```

Leave the options at their defaults.

```
# make install clean
```

We need to install the latest version of the GDB debugger:

```
# cd /usr/ports/devel/gdb
# make config-recursive
```

Leave the options at their defaults.

```
# make install clean
```

Step 4. Install Eclipse Indigo

Install Java

```
# cd /usr/ports/java/openjdk6
# make config-recursive
```

Disable "TZUpdate" in the options for OpenJDK6 port. Leave the rest of the options at their defaults.

```
# make install clean
```

Install Eclipse 3.7 Indigo

```
# cd /usr/ports/java/eclipse
# make config-recursive
```

Leave the options at their defaults.

```
# make install clean
```

If you decide to install Eclipse Juno, then you should install `java/eclipse-devel`.

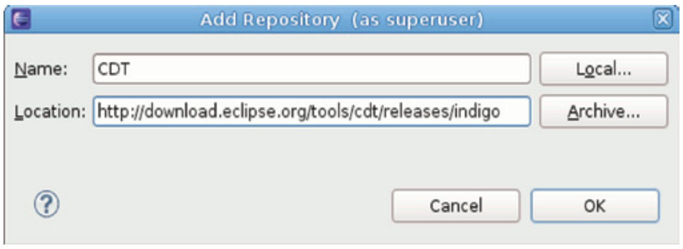
Install CDT

CDT is an Eclipse plugin for C++ development. If we install it, we will be able to use Eclipse for C++ projects.

Run Eclipse as root:

```
# sudo /usr/local/bin/eclipse
```

Choose "Help -> Install New Software". Click "Add". Type "CDT" as the name and *http://download.eclipse.org/tools/cdt/releases/indigo* as the location (or *http://download.eclipse.org/tools/cdt/releases/juno* if you have installed Eclipse Juno from `java/eclipse-devel`):


Add Repository (as superuser)

Name: CDT

Location: http://download.eclipse.org/tools/cdt/releases/indigo

Figure 1. *Add repository*

Click "OK". If this is not possible and you get an error message of a duplicate location, then the link already exists. Click "Cancel" and, back in the previous dialog, click on "Available Software Sites".

Select all the options from the "CDT Main Features" and the "CDT Optional Features".

Click "Next" twice, accept license, and click "Finish".

Exit Eclipse. Now if you run Eclipse under your user, it should have the C++ perspective and you will be able to create C++ projects.

Step 5. Enable shared folders with your Windows host

Share a folder on Windows host

I use Microsoft Windows 7 Professional 64 bit as my host machine. The procedure may differ on other versions of Microsoft Windows.

Right click on a folder, and then select "Properties" from the popup menu. Click on the "Sharing" tab and click on the "Advanced Sharing" button.

Enable sharing ("Share this folder" checkbox) and click on the "Permissions" to disable sharing for "everyone" and allow your Microsoft Windows user to read/write the folder.

Configure FreeBSD guest

Add the following lines to `/etc/nsmb.conf`:

```
[WIN_HOST]
addr=<your_win_ip_address>
[WIN_HOST:WIN_USERNAME]
password=<your_win_password>
```

Please note that WIN_HOST and WIN_HOST:WIN_USERNAME should be in capital letters.

Add the following line to `/etc/fstab`:

```
//WIN_USERNAME@WIN_HOST/WIN_SHARED_FOLDER   FREEBSD_FOLDER
   smbfs rw,auto,-f=770,-d=770,-u=your_freebsd_username,-
   g=wheel 0 0
```

Change firewall rules on your Microsoft Windows host to allow connections from your FreeBSD guest. Reboot and enjoy.

Step 6. Configure your environment to support C++11

```
# cd /usr/local/bin/
# ln -s gcc48 gcc
# ln -s g++48 g++
```

Add the following lines to `/etc/profile`:

```
export PATH=/usr/local/bin:${PATH}
export LD_LIBRARY_PATH=/usr/local/lib/gcc48:${LD_LIBRARY_PATH}
export CC=/usr/local/bin/gcc48
export CXX=/usr/local/bin/g++48
alias make="gmake"
alias gdb="/usr/local/bin/gdb76"
alias g++="g++48"
```

Create for your user (not root) the file `~/.bashrc`:

```
source /etc/profile
```

Change default shell to bash:

```
# chsh -s /usr/local/bin/bash your_username
```

Reboot.

Creating a C++ project from an existing MakeFile

When creating a C++ project from an existing MakeFile, you should choose "Linux GCC" as "Toolchain for Indexer Settings". You can always change this setting through the project properties menu: Project -> Properties -> C/C++ Build -> Tool Chain Editor.

I usually use `gmake` so I need to change the following setting: Project -> Properties -> C/C++ Build. Change "Build command" to `/usr/local/bin/gmake`. And also make sure that your "Build directory" points to the directory with your make file.

Project -> Properties -> C/C++ Build -> Discovery options. Add -std=c++11 to "Compiler invocation arguments" and change `g++` to `g++48` in the "Compiler invocation command".

Project -> Properties -> C/C++ Build -> Build Variables. Add variable LD_LIBRARY_PATH with value `/usr/local/lib/gcc48`.

In Debug Configurations on the "Debugger" tab, change debugger to `/usr/local/bin/gdb76`.

Step 7. Install Valgrind

Install Valgrind

Valgrind is a system for debugging and profiling UNIX programs. With the tools that come with Valgrind, you can automatically detect many memory management and threading bugs, avoiding hours of frustrating bug-hunting, making your programs more stable.

Let's install it:

```
# cd /usr/ports/devel/valgrind
# make config-recursive
```

Leave the options at their defaults.

```
# make install clean
```

Install devel/kcachegrind

KCachegrind is a visualization tool for the profiling data generated by Cachegrind (which is part of `devel/valgrind`).

```
# cd /usr/ports/devel/kcachegrind
# make config-recursive
```

Leave the options at their defaults. Make sure that the "DOT" option is selected.

```
# make install clean
```

Profiling with gprof

- Compile your application with the –pg option.
- Run your application as usual. A new file with a .gmon extension will be created.
- Download and save somewhere `gprof2dot.py` from *http://code.google.com/p/jrfonseca/wiki/Gprof2Dot*
- Run the following command:

```
% gprof your_app your_app.gmon | ./gprof2dot.py | dot
                -Tpng -o out.png
```

The file `out.png` will contain your profiling results in a human-friendly manner.

Profiling with valgrind

- Run your application:

```
% valgrind --demangle=yes --tool=callgrind your_app
```

A file, `callgrind.out.<pid>`, will be created.

- Generate a function-by-function summary from the profile data file:

```
% callgrind_annotate callgrind.out.<pid> > out.txt
```

- Download and save somewhere `gprof2dot.py` from *http://code.google.com/p/jrfonseca/wiki/Gprof2Dot*
- Create out.png with profiling data:

```
% ./gprof2dot.py --format=callgrind callgrind.out.<pid> |
                dot -Tpng -o out.png
```

You can also use `KCacheGrind` to review profiling results.

Summary

As you can see from this article all necessary tools for C++11 programming exist in the FreeBSD ports tree. That means that anyone can quickly and efficiently create applications for this extremely fast, stable and secure operating system.

ANDREY VEDIKHIN

Andrey Vedikhin aka vand777 has been passionate about technology and computers since his early teens. He works for a financial institution in the UK. He likes to play with FreeBSD in his spare time managing approximately 10 FreeBSD operating system instances running name servers, mail servers, web servers, application servers, database servers, firewalls, etc. for some corporate and personal domains on the Internet. His hobby is C++ programming for UNIX, Linux and Windows operating systems.

FreeBSD Programming Primer – Part 9

In the ninth part of our series on programming, we will add some security to our CMS and refine our interface for adding content.

What you will learn…

- How to to configure a development environment and write HTML, CSS, PHP and SQL code

What you should know…

- BSD and general PC administration skills

In part 8 of the series, we created the PHP script amendcontentpage.php which allowed the user to add data to any table in the CMS (Figure 1 & 2). We will refine this page, and add a login page and the corresponding database table. Create a login table in MySQL to hold the user credentials (Listing 1).

We require a blob field as we will be storing binary rather than string data for the encrypted password. The auth field will be used to define the user rights later on, but for the moment setting a value of 1 via the form we will construct will be sufficient.

Now add the following to our global.css file to format the output from our amendcontent.php page (Listing 2).

Create a file in the includes directory called login.inc (Listing 3). This holds the name and secret key for the login cookie.

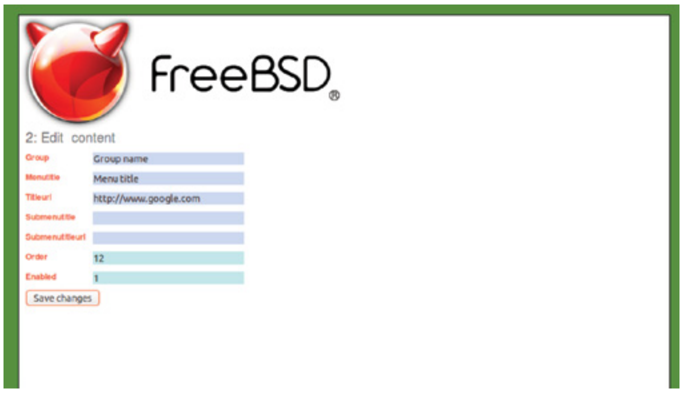
Create a new file login.php in the root directory of the application (Listing 4).

Finally, amend amendcontent.php as follows. As there are a lot of changes throughout the file, the script is detailed here in its entirety (Listing 5).

Hopefully, most of the code should be self explanatory, but here is a breakdown of the major functionality of each page.

Login.php

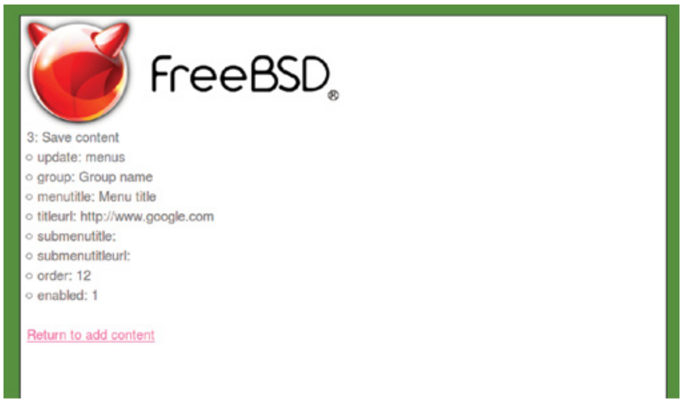
As we are storing the hashed value of the password in the database, rather than a text string that we can com-



2: Edit content

Group	Group name
Menutitle	Menu title
Titleurl	http://www.google.com
Submenutitle	
Submenutitleurl	
Order	12
Enabled	1

Figure 1. *Select the table to edit*



3: Save content
- update: menus
- group: Group name
- menutitle: Menu title
- titleurl: http://www.google.com
- submenutitle:
- submenutitleurl:
- order: 12
- enabled: 1

Figure 2. *Saving the data*

Listing 1. *Create a login table in MySQL*

```
CREATE TABLE `login` (
  `id` int(5) unsigned zerofill NOT NULL AUTO_INCREMENT,
  `username` varchar(25) NOT NULL,
  `password` blob NOT NULL,
  `auth` int(1) NOT NULL,
  `timestamp` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=latin1;
```

Listing 2. *Additions to global.css*

```
.tablehdr {
    background-color: rosybrown;
    color: white;
    float: left;
    font-size: 14px;
    font-weight: bold;
    line-height: 25px;
    padding: 10px;
    width: 22%;
}
.tablerow1 {
    background-color: thistle;
    float: left;
    font-size: 14px;
    line-height: 25px;
    padding: 10px;
    width: 22%;
}
.tablerow2 {
    background-color: oldlace;
    float: left;
    font-size: 14px;
    line-height: 25px;
    padding: 10px;
    width: 22%;
}
```

Listing 3. *login.inc*

```
<?php

define('KEYNAME','gpl9867fghlls');
define('LOGINKEY','117hkJ23230rT');
```

Listing 4. *login.php*

```
<?php

require_once 'includes/cms.inc';
require INCLUDES . 'content.inc';
require INCLUDES . 'core.inc';
require INCLUDES . 'html.inc';
require INCLUDES . 'mysql.inc';
require INCLUDES . 'login.inc';

// SQL statements

$sql[0] = "SELECT password,auth FROM login
           WHERE username = '---P0---'
           AND password = '---P1---';";
$sql[1] = "INSERT INTO `login` (`username`, `password`,
   `auth`, `timestamp`)
         VALUES ('---P0---', ('---P1---'), '---P2---',
   now());";



//////////////////////////////////////////////////
//////////////////////////////////////////////////
//////////////////////////////////////////////////

// Delete these 3 lines after first user added

if(!isset($_POST["action"])){
   createnewlogin();
}

//////////////////////////////////////////////////

// Page control logic

if(isset($_POST["action"])){

    $action = $_POST["action"];

    if($action == "validatelogin"){

      if(isset($_POST["username"]) && isset($_
    POST["password"])){

        $username = $_POST["username"];
        $password = $_POST["password"];

        // We have valid credentials, validate

        validatelogin($username,
    $password,$sql);

      }

    }elseif($action == "createnewlogin"){

      if(!isset($_COOKIE[KEYNAME])){

        // Create a new login to the system

        createnewlogin($username, $password,$sql);

      }else{

        // User failed cookie test, request them to
    login

        requestlogindetails();
      }

    }elseif($action == "appendnewlogin"){

      $username = $_POST["username"];
      $password = $_POST["password"];
      $auth = $_POST["auth"];

      appendnewlogin($username,$password,$auth,$sql);

    }else{

      // Invalid action - request login details

      requestlogindetails();
    }

  }else{

    // First visit to page

    requestlogindetails();
```

```
}

///////////////////////////////////////////////////
///////////////////////////////////////////////////
///////////////////////////////////////////////////

function validatelogin($username, $password, $sql){

  // As the password is hashed and hopefully cannot be
  decrypted,
  // We need to usend the encrypted password

  $hashed_password = hash('whirlpool', $password);

  // Fetch credentials from DB, if match create a login
  cookie

  $s = $sql[0];
  $s = str_replace ( '---P0---' , $username , $s );
  $s = str_replace ( '---P1---' , $hashed_password , $s
  );

  $result = mysql_fetchrows($s);

  foreach($result as $row){

    $auth = $row[1];

  }

  if ($auth == 1) {

    // Create auth cookie

    setcookie(KEYNAME, LOGINKEY, time()+3600, "/");

    // Display options

    $title = 'Welcome ' . $username;

    buildheader($title);
    echo wraptag("h1",$title);
    echo ahref('Add or amend content', '/amendcontent.
  php');
    buildfooter();

  }else{

    // Try again
```

Faster.
Better.
Reliable.

Trusted by over 500 ISPs worldwide.

Hyper is the first multimedia cache fully developed in Brazil, by Taghos.
With Hyper, ISPs can save on network bandwidth while increasing content-delivery speeds, resulting in end-customer satisfaction.

Features:

- 24x7x365 always-on support
- Active monitoring
- Automatic updates
- Appliance or license
- Easy deployment
- Configuration and reports via web interface

hyper

Remote Install
Using your hardware

Model	Traffic	RAM	Cache	SSD
T15	Up to 15 Mbps	8 GB	1x 1 TB	-
T50	Up to 50 Mbps	8 GB	2x 1 TB	-
T100	Up to 100 Mbps	8 GB	2x 1 TB	1x 160 GB
T150	Up to 150Mbps	16 GB	3x 2 TB	1x 160 GB
T300	Up to 300 Mbps	16 GB	5x 2 TB	1x 240 GB
T500	Up to 500 Mbps	32 GB	7x 2 TB	1x 480 GB
T1000	Up to 1 Gbps	64 GB	10x 1 TB	1x 480 GB
T2000	Up to 2 Gbps	96 GB	24x 1 TB	3x 480 GB
T3000	Up to 3 Gbps	128 GB	32x 1 TB	5x 480 GB

Visit us at www.taghos.com and start saving bandwidth today!

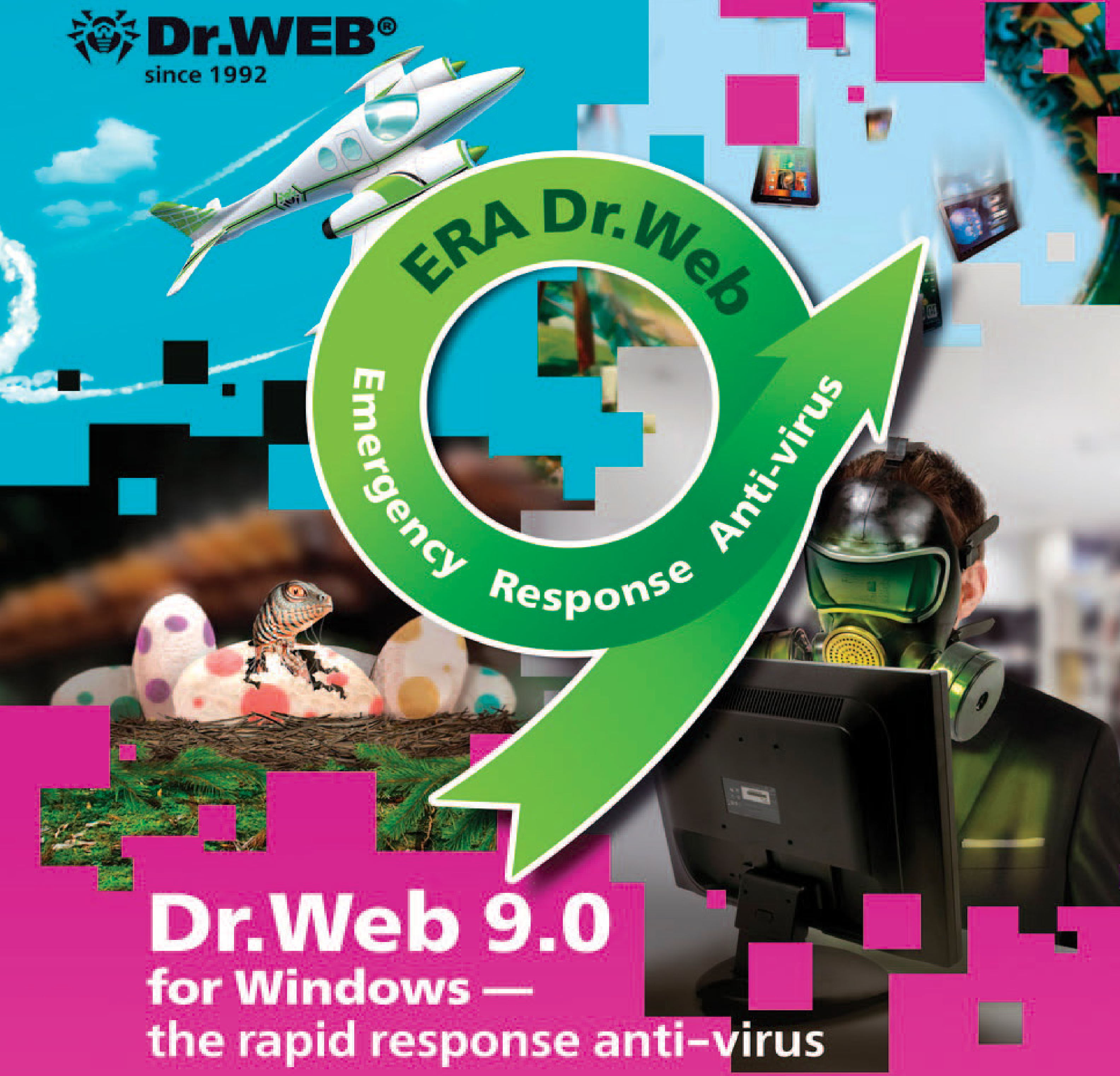
```
        requestlogindetails();

    }

}

function createnewlogin(){

    $title = "Create new user";
    $class = "formcontrol";

    buildheader($title);
    echo wraptag("h1",$title);

    echo '<form action="login.php" method="post">';
    echo 'Username' . div('<input type="text"
    name="username">',$class);
    echo 'Password' . div('<input type="password"
    name="password">',$class);
    echo 'Auth' . div('<input type="text"
    name="auth">',$class);
    echo '<input type="submit" value="Submit">';
    echo '<input type="hidden" name="action"
    value="appendnewlogin">';
    echo '</form>';

    buildfooter();

}

function appendnewlogin($username,$password,$auth,$sql){

    // Create a new entry in the login table

    $hashed_password = hash('whirlpool', $password);

    $s = $sql[1];
    $s = str_replace ( '---P0---' , $username , $s );
    $s = str_replace ( '---P1---' , $hashed_password , $s
    );
    $s = str_replace ( '---P2---' , $auth , $s );

    mysql_select($s);
    requestlogindetails();

}

function requestlogindetails(){

    $title = "Please login";
    $class = "forminput";

    buildheader($title);

    echo wraptag("h1",$title);
    echo '<form action="login.php" method="post">';
    echo 'Username' . div('<input type="text"
    name="username">',$class);
    echo 'Password' . div('<input type="password"
    name="password">',$class);
    echo '<input type="submit" value="Submit">';
    echo '<input type="hidden" name="action"
    value="validatelogin">';
    echo '</form>';

    buildfooter();


}

function buildheader($title){

    // As cookies need to be set before any output is
    sent to the browser
    // use a function call to build the page header

    // Build the page up to the body tag

    outfile(TEMPLATES . 'header.inc');

    echo wraptag('title', $title);
    echo HEAD;
    echo BODY;
    echo '<div id="content">';
    echo '<div id="php">';

}

function buildfooter(){

echo '</div>';
echo '</div>';
echo '<div id="licence">';
echo '<a href="licence.txt" title="Copyright and licence
    details">Copyright &copy; 2013 Rob Somerville me@
    merville.co.uk</a>';
echo '</div>';
```

Listing 5. *amendcontent.php*

```
<?php

require_once 'includes/cms.inc';
require INCLUDES . 'content.inc';
require INCLUDES . 'core.inc';
require INCLUDES . 'html.inc';
require INCLUDES . 'mysql.inc';

// SQL statements

$sql[0] = "SELECT COUNT(DISTINCT TABLE_NAME) FROM
    INFORMATION_SCHEMA.COLUMNS
            WHERE table_schema = 'freebsdcms'
            AND TABLE_NAME = '---P0---'";
$sql[1] = "SELECT TABLE_NAME,COLUMN_NAME,COLUMN_
    DEFAULT,IS_NULLABLE,DATA_TYPE,CHARACTER_MAXIMUM_
    LENGTH
            FROM INFORMATION_SCHEMA.COLUMNS
            WHERE table_schema = 'freebsdcms'
            AND TABLE_NAME = '---P0---'
            ORDER BY table_name, ordinal_position";
$sql[2] = "SELECT * FROM ---P0--- ORDER BY id DESC";
$sql[3] = "SELECT COLUMN_NAME FROM INFORMATION_SCHEMA.
    COLUMNS WHERE TABLE_SCHEMA = 'freebsdcms' AND TABLE_
    NAME = '---P0---'";
$sql[4] = "SELECT `---P0---` FROM ---P1--- WHERE id ='--
    -P2---'";

// The tables we will allow the user to edit via this
    form

$tables[] = "faqs";
$tables[] = "menus";
$tables[] = "news";
$tables[] = "pages";

// Fields that are automatically assigned via a default
    value in MySQL table definition

$skiplist[] = "id";
$skiplist[] = "timestamp";

///////////////////////////////////////////////////
///////////////////////////////////////////////////
///////////////////////////////////////////////////

// Build the page up to the body tag

outfile(TEMPLATES . 'header.inc');

echo wraptag('title', 'Content Input');
echo HEAD;
echo BODY;

// Page control logic

if(isset($_POST["table"])){

     // User has not selected a table or we are testing
    their result

     $t = $_POST["table"];

     if (!in_array($t, $tables)) {

          // If the table is not on allowed list, bail to
    the first page

          build_page_1($tables);

     }else{

          // Check selected table is valid

          $s = $sql[0];

          // Replace the marker in the SQL statement with
    the chosen value
          $s = str_replace ( '---P0---' , $t , $s );

          $result = mysql_select($s);
          $valid_table_count = $result['COUNT(DISTINCT
    TABLE_NAME)'];

          if($valid_table_count == 1){

             // Valid table selected - present form to edit data
              build_page_2($t,$sql,$skiplist);

          }else{

              // Send user to first page
              build_page_1($tables);

          }

     }
```

```
}elseif(isset($_POST["update"])){

    // Save the input. As we have not validated this,
   just display for now

    build_page_3($_POST);

}elseif(isset($_POST["id"])){

    // Save the input. As we have not validated this,
   just display for now

    echo "Updateold record";

}else{

    // Invalid value - return to start

    build_page_1($tables);

}

///////////////////////////////////////////////////
///////////////////////////////////////////////////
///////////////////////////////////////////////////

function build_page_1($tables){

// HTML form definition

echo '<div id="content">';
echo '<div id="php">';
echo '<div id="h1">Select content</div>';
echo '<form action="amendcontent.php" method="post">';

// Build the list of tables

$tablecontrol = '';
$tablecontrol .= '<select name="table">';

foreach ($tables as $t){

  // $tables is an array - split each value out

  $tablecontrol .= '<option value="'.$t.'">'.$t.'</option>';

}
$tablecontrol .= '</select>';

// Build the edit options

$editcontrol = '';
$editcontrol .= '<input type="radio" name="inputmode"
   value="new" checked="checked">Add new content';
$editcontrol .= '<input type="radio" name="inputmode"
   value="update">Update current content';

// Build the submit option

$submitcontrol = '';
$submitcontrol .= '<input type="submit" value="Create
   content">';

// Add the DIV to format the controls

echo div($tablecontrol, 'formcontrol');
echo div($editcontrol, 'formcontrol');
echo div($submitcontrol, 'formcontrol');

// Complete the form

echo '</form>';
echo '</div></div>';
echo '<div id="licence">';
echo '<a href="licence.txt" title="Copyright and licence
   details">Copyright &copy; 2013 Rob Somerville me@
   merville.co.uk</a>';
echo '</div>';

}

function build_page_2($t,$sql,$skiplist){

// HTML form

echo '<div id="content">';
echo '<div id="php">';

// Check we have a valid inputmode

if(!isset($_POST["inputmode"])){

    echo "Error: Invalid inputmode";
    exit;

}else{
```

Dr.WEB®
since 1992



Dr.Web 9.0
for Windows — the rapid response anti-virus

1. Reliable protection against the threats of tomorrow
2. Reliable protection against data loss
3. Secure communication, data transfer and Internet search

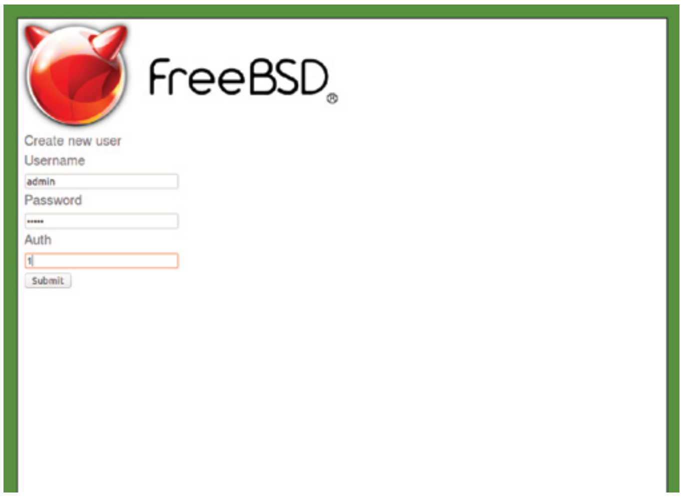
www.drweb.com
Free 30-day trial: https://download.drweb.com
New features in Dr.Web 9.0 for Windows: http://products.drweb.com/9
FREE bonus — Dr.Web Mobile Security: https://download.drweb.com/android

```
if($_POST["inputmode"] == "new" || (isset($_
   POST["rowid"]))){

     // New content - populate with selected value if we
   have arrived here
     // via an update content request.

     if(isset($_POST["rowid"])){

       $rowid = $_POST["rowid"];
       $populate = TRUE;

     }else{

       $rowid = "";
       $populate = FALSE;

     }


     echo '<div id="h1">Create new '.$t.' content</
   div>';
     echo '<form action="amendcontent.php"
   method="post">';

     // Get the schema for that particular table

     $s = $sql[1];
     $s = str_replace ( '---P0---' , $t , $s );

     $result = mysql_fetchrows($s);
     $divstart = '<div class="inputname">';
     $action = 'Save';

     echo '<input type="hidden" name="update"
   value="'.$t.'">';

     foreach($result as $row){

     // Loop through each field and build the form fields
   depending on the field type

        $field =  $row[1];
        $fieldtype =  $row[4];

       if (!in_array($field, $skiplist)) {

            if($fieldtype == "varchar"){

                $value = populatefields($sql[4],$field,$t,$row
id,$populate);

                echo  $divstart . ucfirst($field).'</div><input
class="varchar" type="text" name="' .$field. '"
value="'.$value.'"><br />';

                }elseif($fieldtype == "int"){

                 $value = populatefields($sql[4],$field,$t,$ro
wid,$populate);

             echo  $divstart . ucfirst($field).'</div><input
class="int" type="text" name="' .$field. '"
value="'.$value.'"><br />';

                }elseif($fieldtype == "text"){

                    $value = populatefields($sql[4],$field,$t,
$rowid,$populate);

                    echo  $divstart . ucfirst($field).'</
div><textarea rows="10" cols="30" class="textarea"
name="' .$field. '">'.$value.'</textarea><br />';

                }else{

                    // Shouldn't get here

                 echo 'Error field('.$field.')  '.
$row[2].'|'. $row[3] .'|'.$row[4].'|'. $row[5] .'<br
/>';

                }

           }

       }

       //echo '</select>';

       }elseif($_POST["inputmode"] == "update"){

        echo '<div id="h1">Select content '.$t.'</div>';
        echo '<form action="amendcontent.php"
method="post">';
        echo '<input type="hidden" name="table"
value="'.$t.'">';
```

```
  echo '<input type="hidden" name="inputmode"
value="new">';

  $s = $sql[3];

  // Replace the marker in the SQL statement with the
chosen value

  $s = str_replace ( '---P0---' , $t , $s );

  if($t == 'menus'){

   // DB schema is different
   // NB: Maximum cols = 3 unless mods to CSS
performed

   $displaycols = array(2, 4, 5);

  }else{

   // Everything else

   $displaycols = array(1, 2, 3);

  }

  // Get the field names for our table

  $titles = mysql_fetchrows($s);

  // Build the title row

  echo div('Select', 'tablehdr');

  foreach ($displaycols as $offset) {

     echo div($titles[$offset][0], 'tablehdr');

  }

  $s = $sql[2];
  $zebra = 0;
  $action = 'Update';

  // Replace the marker in the SQL statement with the
chosen value
  $s = str_replace ( '---P0---' , $t , $s );

  $result = mysql_fetchrows($s);
```

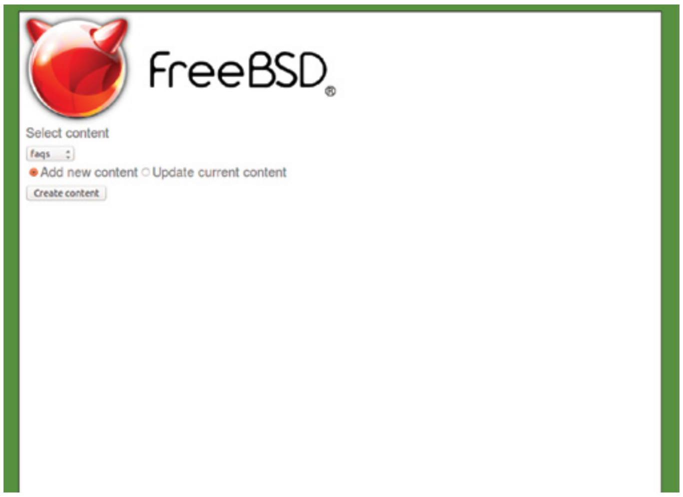
```
     foreach($result as $row){

       if($zebra == 0){

        $class = 'tablerow1';
        $zebra = 1;

       }elseif($zebra == 1){

         $class = 'tablerow2';
         $zebra = 0;

       }

       // Radio button control

       $editcontrol = '<input type="radio" name="rowid"
   value="'.$row[0].'">';

        // Check formatting and output

       formatcontentedit($row, $class, $displaycols,
   $editcontrol);

     }

    }else{

        echo "Error: Invalid inputmode";
        exit;

    }

}

// Finish form and add footer

echo '<input type="submit" value="'.$action.' '.$t.'
   item">';
echo '</form>';
echo '</div></div>';
echo '<div id="licence">';
echo '<a href="licence.txt" title="Copyright and licence
   details">Copyright &copy; 2013 Rob Somerville me@
   merville.co.uk</a>';
echo '</div>';

}
```

```
function build_page_3($post){

// HTML

echo '<div id="content">';
echo '<div id="php">';
echo '<div id="h1">Save content</div>';
echo '<ul>';

foreach($post as $key => $value){

    // Just dump out values - we need to validate before
   adding to DB

    echo '<li><b>'.$key.'</b>: '.$value.'</li>';

}

// End of form

echo '</ul><br />';
echo '<a href="amendcontent.php">Return to add content</a>';

echo '</div></div>';
echo '<div id="licence">';
echo '<a href="licence.txt" title="Copyright and licence
   details">Copyright &copy; 2013 Rob Somerville me@
   merville.co.uk</a>';
echo '</div>';

}

function formatcontentedit($row, $class, $displaycols,
   $editcontrol){

    // Formats the rows from our select query in zebra
   format.
    // To prevent the CSS from breaking due to NULL
   content
    // and displays the appropriate rows as the menu
   schema is
    // different from everything else.

    // Display the radio button

    echo div($editcontrol, $class);

    // Format each row

    foreach ($displaycols as $offset) {

        // First check we have some content - use a NBSP
    if NULL or blank

        if($row[$offset] == ''){

            $row[$offset] = '&nbsp;';

        }

        // Ensure length < 25 chars, else add elipses

        if(strlen($row[$offset]) > 25){

            $row[$offset] = substr($row[$offset],0,24) .
    ' ...';

        }

        // Display each field from the row

        echo div($row[$offset], $class);

    }
}

function populatefields($sql,$field,$t,$rowid,$populate){

    if($populate){

      $s = str_replace ( '---P0---' , $field , $sql );
      $s = str_replace ( '---P1---' , $t , $s );
      $s = str_replace ( '---P2---' , $rowid , $s );

      $v = mysql_fetchrows($s);

      $value = $v[0][0];

    }else{

      $value = "";

    }

    return $value;

}
```



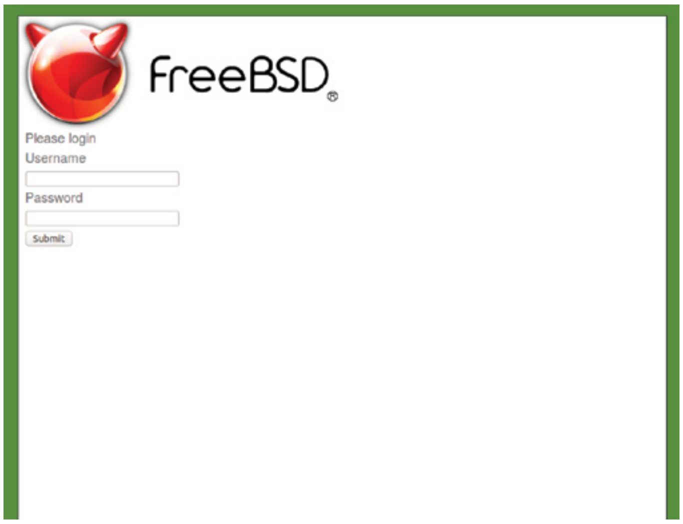
Create new user
Username
admin
Password
•••••
Auth
1

Figure 3. *Creating a new user*



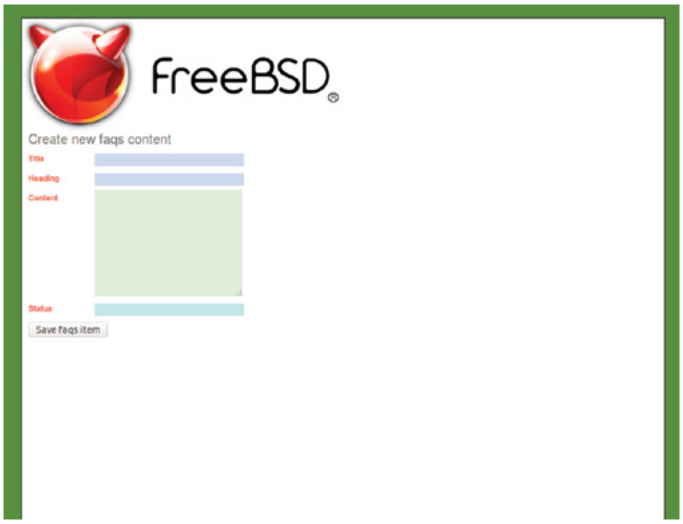
Select content
faqs
[✓ Add new content] Update current content

Figure 6. *Choose your content type add new or update*



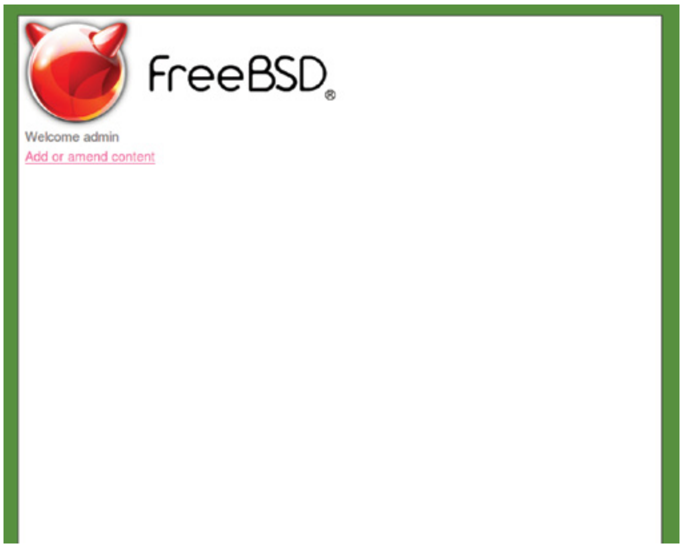
Please login
Username
Password

Figure 4. *Logging in*



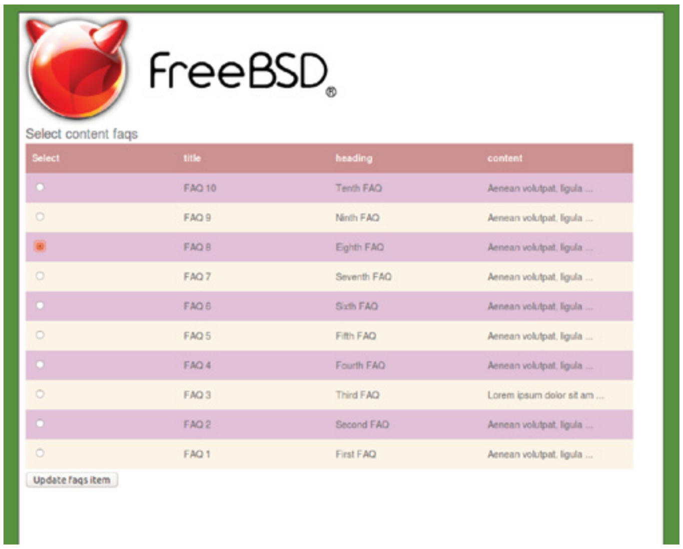
Create new faqs content
Title
Heading
Content
Status

Figure 7. *Adding a new FAQ*



Welcome admin
Add or amend content

Figure 5. *Add or amend content*



Select content faqs

Select	title	heading	content
○	FAQ 10	Tenth FAQ	Aenean volutpat, ligula ...
○	FAQ 9	Ninth FAQ	Aenean volutpat, ligula ...
●	FAQ 8	Eighth FAQ	Aenean volutpat, ligula ...
○	FAQ 7	Seventh FAQ	Aenean volutpat, ligula ...
○	FAQ 6	Sixth FAQ	Aenean volutpat, ligula ...
○	FAQ 5	Fifth FAQ	Aenean volutpat, ligula ...
○	FAQ 4	Fourth FAQ	Aenean volutpat, ligula ...
○	FAQ 3	Third FAQ	Lorem ipsum dolor sit am ...
○	FAQ 2	Second FAQ	Aenean volutpat, ligula ...
○	FAQ 1	First FAQ	Aenean volutpat, ligula ...

Figure 8. *Choosing an existing FAQ to edit*

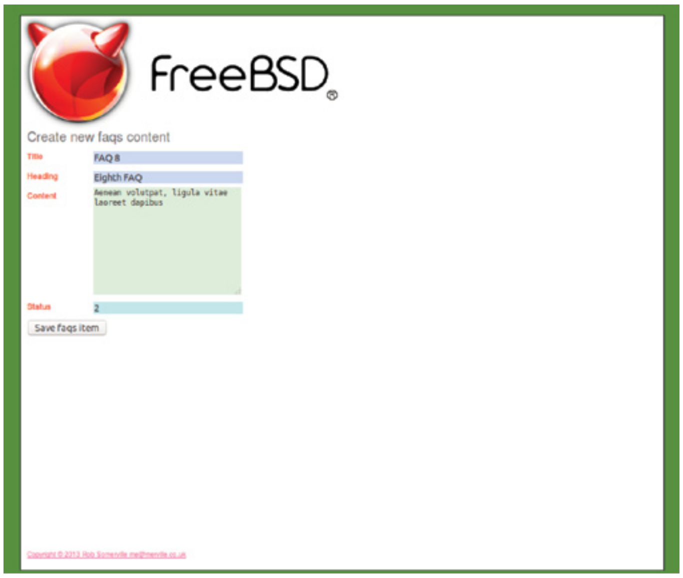
pare, we need to initially seed the database with a valid username and password the first time login.php is run. Once the login table is updated, the call to `createnewlogin()` can be removed. The page control logic branches depending on the action we want to achieve, and the corresponding function calls either build an HTML form, query the database or add a user to the database.

Amendcontent.php

Most of the action takes place within `build_page_2`. In the previous version, we could not select any previously entered content so to add this functionality we have added an intermediate step which displays all the content available to be edited. Once the user selects the table record to be amended, this is fed back into our original form, which is essentially identical whether we are updating or adding content. Some "bells and whistles" are added in the form of zebra striping of the table rows, and the automatic generation of the titles. As the script is referring directly to the database, we need a conditional branch at line 266 as the menu table has a different schema from the pages, news and FAQ content.

Getting it to work

Visit the login item via the menu and create a new user and password with an auth level of 1. Check the user has been

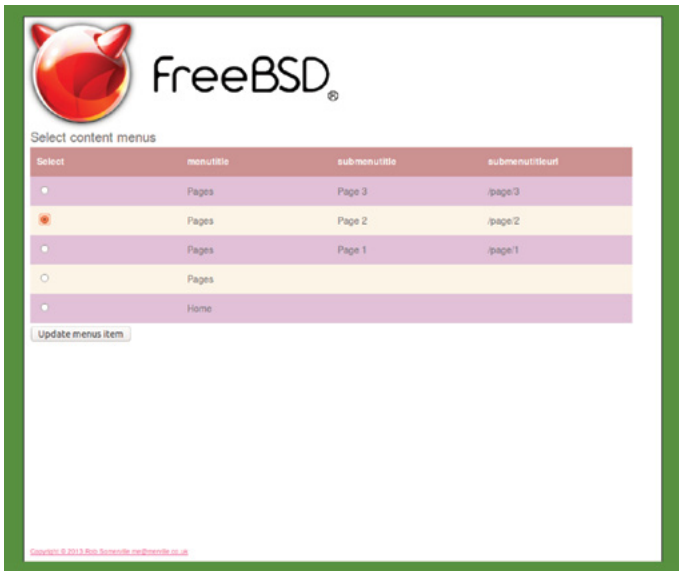


Create new faqs content

Title	FAQ 8
Heading	Eighth FAQ
Content	Aenean volutpat, ligula vitae laoreet dapibus
Status	2

Copyright © 2013 Rob Somerville me@merville.co.uk

Figure 9. *Editing a pre-existing FAQ*

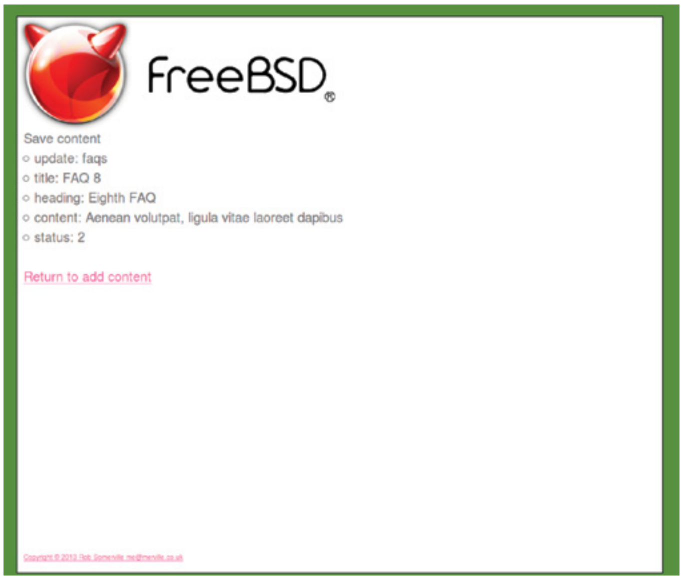


Select content menus

Select	menutitle	submenutitle	submenutitleurl
○	Pages	Page 3	/page/3
●	Pages	Page 2	/page/2
○	Pages	Page 1	/page/1
○	Pages		
○	Home		

Copyright © 2013 Rob Somerville me@merville.co.uk

Figure 11. *Picking an existing menu to edit*



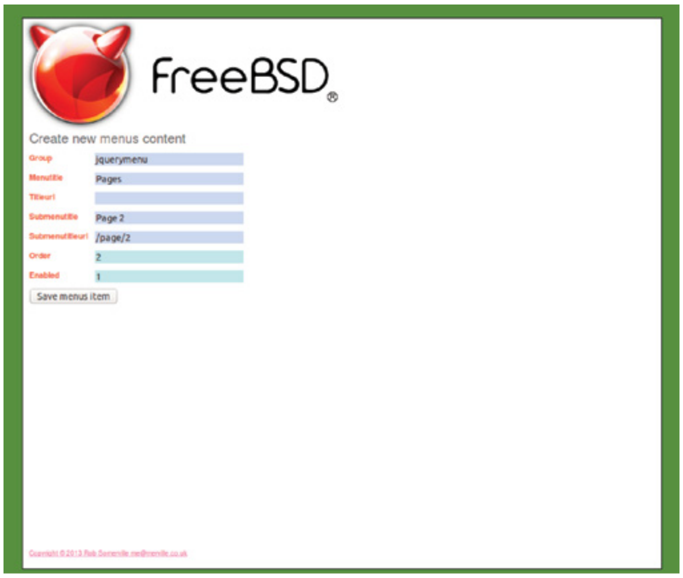
Save content

- update: faqs
- title: FAQ 8
- heading: Eighth FAQ
- content: Aenean volutpat, ligula vitae laoreet dapibus
- status: 2

Return to add content

Copyright © 2013 Rob Somerville me@merville.co.uk

Figure 10. *Saving the FAQ*



Create new menus content

Group	jquerymenu
Menutitle	Pages
Titleurl	
Submenutitle	Page 2
Submenutitleurl	/page/2
Order	2
Enabled	1

Copyright © 2013 Rob Somerville me@merville.co.uk

Figure 12. *Saving a menu item*

Useful links

PHP manual – *http://php.net/manual*

added to the login table and remove the 3 lines from the start of login.php as commented. Revisit login.php, login and proceed to edit your content as desired (Figure 3-12).

Further tuning

The security is poor – we can have multiple users with the same name and password. A better form of encryption other than hashing is desirable, and we are missing lots of backlinks etc. We should also refactor the code e.g. the license and footers.

In the next part

We will address these issues and more.

ROB SOMERVILLE

Rob Somerville has been passionate about technology since his early teens. A keen advocate of open systems since the mid-eighties, he has worked in many corporate sectors including finance, automotive, airlines, government and media in a variety of roles from technical support, system administrator, developer, systems integrator and IT manager. He has moved on from CP/M and nixie tubes but keeps a soldering iron handy just in case.

FreeBSD

BSD Certification

The BSD Certification Group Inc. (BSDCG) is a non-profit organization committed to creating and maintaining a global certification standard for system administration on BSD based operating systems.

WHAT CERTIFICATIONS ARE AVAILABLE?

BSDA: Entry-level certification suited for candidates with a general Unix background and at least six months of experience with BSD systems.

BSDP: Advanced certification for senior system administrators with at least three years of experience on BSD systems. Successful BSDP candidates are able to demonstrate strong to expert skills in BSD Unix system administration.

WHERE CAN I GET CERTIFIED?

We're pleased to announce that after 7 months of negotiations and the work required to make the exam available in a computer based format, that the BSDA exam is now available at several hundred testing centers around the world. Paper based BSDA exams cost $75 USD. Computer based BSDA exams cost $150 USD. The price of the BSDP exams are yet to be determined.

Payments are made through our registration website:
https://register.bsdcertification.org//register/payment

WHERE CAN I GET MORE INFORMATION?

More information and links to our mailing lists, LinkedIn groups, and Facebook group are available at our website:
http://www.bsdcertification.org

Registration for upcoming exam events is available at our registration website:
https://register.bsdcertification.org//register/get-a-bsdcg-id

Improved Updates and LTS for OpenBSD

Automated security updates for both OpenBSD packages and the base system are now possible using a new utility called openup. M:Tier company started offering LTS (Long Term Support) for companies and individuals.

What you will learn…

- How OpenBSD is structured.
- How to update OpenBSD system via openup.
- Which new progressive features there are in OpenBSD.

What you should know…

- OpenBSD is issued every 6 months and does not offer any package updates by default.
- Core system updates are in the form of source patches only.

Two years ago, I read an interesting article published somewhere on the web, explaining why OpenBSD has so many missing features compared to Linux and other modern Unix-like systems. There was a list with explanations and proposed solutions. Among others, there was a missing package update utility and lack of LTS versions. The discussion quickly transformed into a flame war between OpenBSD die-hards, underrating those features and condemning the author, and the poor author, defending himself and explaining that was not meant as an attack against the project.

Defending one's favorite OS is a nice thing, but improving it is even better. As I watch M:Tier company working on bringing new helpful features to OpenBSD, I feel we can start crossing out those missing features from the list and watch OpenBSD find its way back among the most well equipped Unixes available.

Step one – stable updates and binpatches

In the spring of 2013, along with the 5.3 release, M:Tier introduced updates of stable packages and binary updates to the OpenBSD kernel using binpatches. Updates for stable packages helped keep your system safe for a longer period of time, and binary updates to the system saved admins a lot of compilation time and extra work. (You can find more explicit information concerning these updates in an article in the May issue of BSDMag.) Updates to the packages worked quite well, *pkg_add just* compared installed packages to those on the server and applied the updates. But, OpenBSD does not consist of packages only. Pkg_add was not able to find system binary patches even if they came in the form of packages, because there were no installed packages to compare it to. For each kernel update, a user had to find the update on the server and install it manually with *pkg_add*. The same applied to the binary patches for the other parts of the basic system, such as web server or Xenocara, a modified Xorg distribution. In FreeBSD, Xorg comes in the form of packages; in OpenBSD, it does not. It is a part of the basic system. Such inconvenience called for a solution.

Step two – new automated updates via openup

In September 2013, a new application called *openup* emerged from the workshop of M:Tier developers (based on Andre Nathan's scripts). The application is available on the project website stable.mtier.org. It is free for everyone to download and use and simplifies all the upkeep of OpenBSD machines. So what is it and what does it do? It is a simple application that uses native OpenBSD *pkg_tools* to find out what updates are available and if the user wishes to provide the security check and downloads and applies the updates. A nice feature is it can either be used manually or automated with cron. If automated with *cron*, it operates similar to modern Linux update service notifications. *Openup* checks whether an update is for an installed package or for the OpenBSD kernel and system and whether it's available and if so, it mails the output to the root. If it's run manually, there are more options:

-K do not check for kernel binpatches (when running non GENERIC)
-S do not check for package signatures
-c check/cron mode (cannot be used with -S and/or -n)
-n dry-run mode, no modification is done

Using *openup* is very easy to use. Below is an example:

```
# openup
===> Checking for openup update
-> ftp -Vo - https://stable.mtier.org/openup | awk -F '='
   '/^_OPENUP_VERSION/ { print $2 }'
===> Installing/updating binpatch(es)
-> pkg_add binpatch53-i386-bgpd binpatch53-i386-kernel
   binpatch53-i386-nginx binpatch53-i386-tftpd
binpatch53-i386-kernel-3.0->4.0: ok
binpatch53-i386-tftpd-1.0: ok
Read shared items: ok
Packages with signatures: 2
===> Updating package(s)
-> pkg_add -u
curl-7.26.0p2->7.26.0p3: ok
Read shared items: ok
Packages with signatures: 1
```

Here we could see *openup* checking if it is the latest version of itself, then checking for the updates, downloading them and installing them. It can be seen that openup was able to check for and find updates to the kernel and to the web server (*nginx*), which in this case, both come as packages, and installed them. Then openup proceeded with regular packages updates. It is interesting though that if the first step would fail, that is, *openup* would be in an obsolete version, the update would stop to prevent the system from possible damage. The user would have to update *openup* first and only then continue.

Warning: *openup* is a rather new application, so further function extensions and change of default behavior due to a very progressive development and testing is very probable.

Step three – LTS

The M:Tier company that promoted OpenBSD solutions in the business world came up with the idea that many *sysadmins* must like. Whether it was a company, or a BSD hobbyist, everyone can subscribe to long term updates. Unfortunately, for those who understand the words *opensource* or *free* to mean *without any costs*, this costs something. Not much, however.

The first six months are supported free of charge for any user. People who do not want to re-install the system after a new OpenBSD update is released, however, can pay Euro 200, and they can continue using their system for another half year and stay up to date.

Many people may think this is a lot of money to spend. The important point is that this money goes to the improvements of the OpenBSD system and, part of it, to the OpenBSD project directly. It is also worth mentioning that key M:Tier developers are part of the core OpenBSD team. This ensures that the development of OpenBSD and M:Tiers apps will go hand-in-hand in bringing the goodies to all of us. So, paying for the LTS support is something similar to buying OpenBSD CDs.

It would very be nice if 12 months of free support was available, but this step is at least a step in the right direction. Possibly we will see M:Tier granting free access to longer periods in the future if their initial trial runs well.

Final word

OpenBSD has been available for a long time and has deserved a very good reputation for its robustness and security-focused approach, but it was criticized in the recent past for obsoleting and not being able to compete with other Unixes, which introduced many important security and functionality features. Now also due to M:Tier, it seems to be catching up again. Package updates have been working perfectly since the Spring and kernel and system updates have been much improved lately. Finally LTS has been introduced. Since OpenBSD resides in the business environment and the business environment is demanding, we can expect more important modern features appearing soon.

On the Web

- *https://stable.mtier.org/*
- *http://www.mtier.org/index.php/news/openup/*
- *http://www.openports.se/*
- *http://www.openbsd.org/*
- *http://www.openunix.eu/*

PETR TOPIARZ

Petr Topiarz has been administering BSD web/mail/file servers and Linux desktops in three small Prague-based companies for the last eight years, and has contributed to BSDMag, since its first issues.

The Computer Says "No"

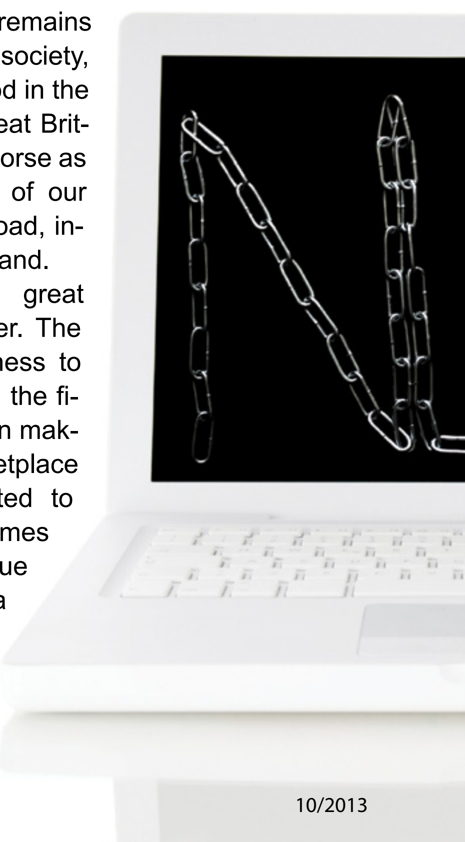
Stanislav Petrov, the officer in command of the Russian Oko monitoring station, prevented global nuclear holocaust by disregarding a false positive alarm and subsequently not reporting this to his superiors in a timely fashion and delaying a knee jerk response. How much reliance can we place on technology and what are the implications for a society where computers are increasingly given ethical responsibility by proxy?

The Turing test is the classic yardstick by which the fragile interface between humans, silicon and the universe is measured. The question "Can computers think?" still causes considerable debate, and while the Turing test is based on the emulation of human characteristics, recent advances in quantum physics- especially quantum entanglement – have demonstrated a more complex universe than we can imagine. According to Bell's theorem, information in our universe travels instantaneously. To simplify, two particles separated by considerable distance will display the same quantum behaviour as if a message has passed instantly between them. This empirically reinforces the argument that a butterfly flapping its wings in one corner of the earth has an effect elsewhere.

That observation aside, I refuse to accept the hypothesis that computers can think on a simple premise – computers do not have a conscience. Without a doubt, computers can fake intelligence, and I would even go as far as to say the argument that computers demonstrate a minute level of consciousness holds some water, provided the definition of consciousness is broad enough (for example being aware of the environment). But again this argument is faux, an emulation and bordering on sophistry. What is deeply concerning though, is the increasing mindless delegation of ethics, morality, decision making and authority to a piece of hot silicon in a data-centre somewhere. If Stanislav Petrov had not had the common sense to compare the false alarm against other criteria, we may not have been around today to have this discussion.

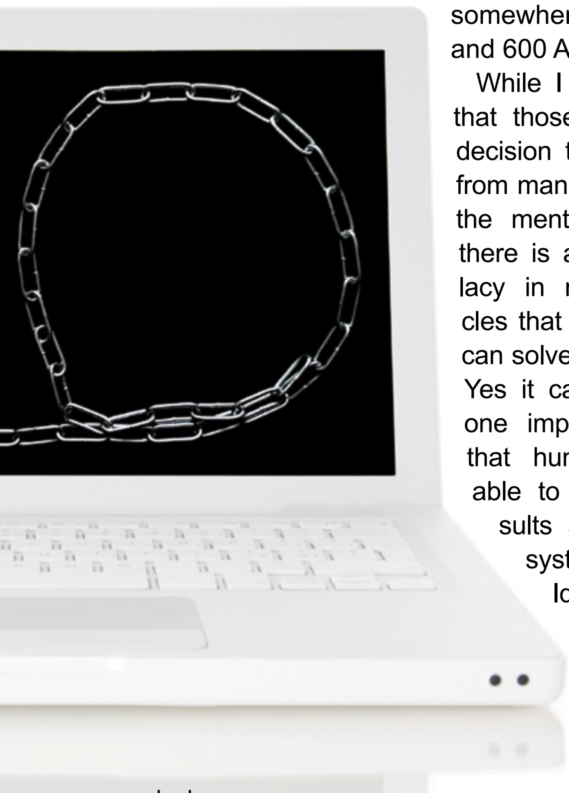
Unfortunately, since this information was released in 1974, as a civilization we have become considerably less aware of the dangers and much more dependent upon technology than ever before. The twin curse of commercial efficiency and profit-making have pushed us over the peak of the bell curve, and there is now no slack in the system for error. Just In Time production is a case in point. While in deeper reaches of the Internet those that defend preparedness are considered "Tin foil hatters", the reality remains that in a modern Western society, we only have 24 hours food in the shops. Where I live in Great Britain, the situation is even worse as a substantial percentage of our food is imported from abroad, including Vietnam and Thailand.

Technology makes a great slave but a terrible master. The past 20 years bears witness to this – with the big bang in the financial sector, the decision making process in the marketplace was increasingly delegated to technology, yet many times the plug has been pulled due to the system developing a form of hysteresis in that they were attempting to interpret new scenarios



using inaccurate data and consequently failing miserably. While the recent financial crash cannot be laid squarely at the foot of technology, it is not a risk free strategy attempting to out-gun your competitors by means of brute speed alone. The recent demand in the financial sector for using higher speed backbones for communications purely on the basis that shaving a few milliseconds per transaction being the difference between a substantial profit or a spectacular loss, should raise alarm bells if that is the best strategy of your business sector.

I am stubborn in my belief that technology has immense benefits to offer mankind, but the more powerful the tool, the greater care one must take in its implementation and use. As a child, my father gave me a plastic tool-kit with plastic spanners, screwdrivers, etc. The havoc reigned with their real life counterparts in the hands of a 5 year old doesn't bear thinking about as the following tale will illustrate. As a young lad I owned a 12 volt Airfix power supply, and gaining access to my father's large metal screwdriver, I discovered that a substantial spark could be generated by shorting out the terminals. With uncontrolled ego and lust for excitement, I decided that the 240v AC emanating from the wall socket would make an interesting test case, and after procuring two 6 inch nails I managed to successfully circumnavigate the safety shutter and insert the aforementioned metallic items into the socket and short them out. The resulting explosion and flash was indeed spectacular, and while both child and screwdriver survived, the nails and the considerably large fuse at the local substation (Rated somewhere between 100 and 600 Amps) did not.



While I am not implying that those who make the decision to wholly migrate from manual systems have the mentality of children, there is a widespread fallacy in management circles that technology alone can solve all the problems. Yes it can help, but with one important proviso – that human beings are able to interpret the results and override the system as necessary. Ideally, technology should be democratic rather than fascist, an educated voice rather than capturing the entire decision making process and expediting the result via a closed interface that cannot be overridden. Thankfully some industries understand this well, most modern trains and passenger aircraft are fly by wire and very safe. However, there are always trained pilots or drivers on hand to take control if the unexpected does happen, and it is a statistical guarantee that it will.

Where the problem arises is when management, the shareholders or whoever decide that "the automated system" will become the de-facto ethical and moral standard – removing the human element and as a consequence any common sense. You can code all the algorithms you like, but you cannot replace the breadth of human experience, understanding and wisdom. Again, the Internet proves this, we have unparalleled access to data but extrapolating the kernel of truth from the propaganda, opinion and noise makes quality information gathering difficult.

The badly designed script driven call centre is probably the best analogy to describe this scenario. Using a set of computer based rules, the operator is constrained by company policy and lack of autonomy – if the computer says "No" there is no chance of appeal, unless of course you manage to capture the ear of a sympathetic manager. Senior management are unaware of the level of gross customer dissatisfaction, as the metrics cannot reflect the moral offence of a human relaying a decision made by a lump of metal, effectively de-humanising the interaction. Rather than improving customer service and efficiency, the opposite occurs shortly followed by a battalion of fire-fighting reputation managers and PR gurus when the wheels fall off.

I propose we name this stupidity in honour of the man who saved countless millions of lives "The anti-Petrov effect".

ROB SOMERVILLE

Rob Somerville has been passionate about technology since his early teens. A keen advocate of open systems since the mid eighties, he has worked in many corporate sectors including finance, automotive, airlines, government and media in a variety of roles from technical support, system administrator, developer, systems integrator and IT manager. He has moved on from CP/M and nixie tubes but keeps a soldering iron handy just in case.

Interview with Klaus P. Ohrhallinger

Please introduce yourself. Tell us, how you've started to be involved in FreeBSD.

My name is Klaus P. Ohrhallinger, I'm 28 years old and live in Europe. I've been using FreeBSD since 4.4, which was released in 2001, but was never involved in the FreeBSD Project back then. My first contacts with FreeBSD developers were when I announced my VPS project in 2010. After that, I got invited to FreeBSD developer summits.

What hides under the curtains of VPS (your container solution)? In other words, what inspired you to develop it?

The most exciting and challenging part is the live-migration feature. When I started development on VPS, I only implemented the absolutely necessary bits and then continued with the live-migration support. I wanted to see whether I could do it. Also I believe that server infrastructure needs to be more efficient, especially in terms of power consumption. Virtualization on the OS level allows you to put a big number of virtual servers on a few physical machines. That means a big change on the ecological impact.

Could you please elaborate on this live-migration feature. What mechanisms are used to perform migration and to sync data?

For performing a live migration of a VPS instance, its state needs to be frozen, first. This means all threads are suspended from execution, the network stack stops accepting any input, and so on. Then lots of tiny bits of information are serialized and saved according to a machine-independent format. For example, the state of open tcp connections, the virtual memory map and objects of a process, filesystem mounts, file descriptors, process relationships. This dump is transferred to another machine (usually through ssh) along with all necessary userspace memory pages. There, sanity checks are performed and new system objects like threads, virtual memory objects, tcp connections are created according to the previously saved information. Execution is resumed exactly where it was suspended in the original instance. Users connected to a VPS instance that is being live migrated will only notice a short network interruption. File system data is synced using a slightly modified version of rsync, unless the filesystem resides on some sort of network storage anyway.

How long did it take you to design and program it?

So far I spent about 1.5 years full-time or 3000 hours on its development.

This number is pretty impressive. You've done it on your own, or some company funded you? If the first is true, how could you estimate this project time, when there would be a special funding campaign?

I got paid by TransIP.NL for one month (160 hours) of development. Apart from that I did not receive any funding. These 3000 hours were spread over 4 years. With proper funding I could have done that without interruptions.

What are the pros and cons (if they exist) for using VPS in production at the moment?

I don't recommend it for production use yet. However I'm already using it for some non-critical production stuff and I hear of companies that are also quite happy with the stability.

Being a developer how could you compare VPS with other technologies, like jail (*BSD) and OpenVZ/LXC (Linux)?

VPS has a very similar design as OpenVZ. Actually Virtuozzo (the commercial and enhanced version of OpenVZ) inspired me a lot and I wanted to have the same for FreeBSD. Jail can't be compared to VPS. It is not a virtualization solution since it only isolates some resources, but does not provide virtual namespaces. It doesn't provide the features that you get with any modern virtualization solution and also it is not safe to provide jails to customers, in my opinion.

Do you know big enterprises / hostings that currently heavily use your solution?

I don't know of any big ones, but as long as everything works fine you hardly get any feedback. I know of some smaller companies that use it for various purposes.

Any chances to run other OSes, for instance, Linux and or Windows with VPS?

Users reported that they installed Debian GNU/kFreeBSD successfully into a VPS instance. Running native Linux distributions in a VPS instance could be possible with limitations. However there is no chance of running Windows.

What are these limitations? When do you plan to program workarounds for those issues?

Mainly that system utilities like, "ifconfig'' will not work. Using their native FreeBSD counterparts, would be a possible work-around. Providing full Linux compatibility would be a very big task, so I don't have any plans for that yet.

Where can VPS currently be found: FreeBSD 9, FreeBSD 10, any other?

The first prototype is based on FreeBSD 8. Alpha-quality releases for FreeBSD 9.0 and 9.1 exist but are not continued anymore. Currently development only happens for the FreeBSD 10 version. It is still not part of the official FreeBSD code base but lives as a project branch in FreeBSD's SVN repository.

So it is sufficient to grab the project's files (please indicate URI), compile it and have fun? No kernel and world rebuilding required?

For FreeBSD 9.x I released binary packages that can be installed via pkg_add. When FreeBSD 10.0 gets released I will publish binary packages for it as well. If you compile it yourself you always need a kernel rebuild. The only changes to the userspace are some new commands, that can be built separately. The project's webpage is at *http://www.7he.at/freebsd/vps/*.

Any web-interfaces exist to control VPS?

There is no web-interface yet, but a non-free management and clustering solution that includes a GUI and a powerful JSON interface. Using that interface, VPS could easily be integrated by hosting providers into their existing web-based control panels and deployment solutions.

Where could a system administrator get support / help including commercial subscription?

Commercial support and custom features can be requested at 7HE Ltd (*office@7he.co.uk*).

ANTON BORISEV

[GEEKED AT BIRTH]

University of Advancing Technology
UAT
Learn. Experience. Innovate.

You can talk the talk.
Can you walk the walk?

[IT'S IN YOUR DNA]

LEARN:
Advancing Computer Science
Artificial Life Programming
Digital Media
Digital Video
Enterprise Software Development
Game Art and Animation
Game Design
Game Programming
Human-Computer Interaction
Network Engineering
Network Security
Open Source Technologies
Robotics and Embedded Systems
Serious Game and Simulation
Strategic Technology Development
Technology Forensics
Technology Product Design
Technology Studies
Virtual Modeling and Design
Web and Social Media Technologies

www.uat.edu > 877.UAT.GEEK

NET OPEN SERVICES IS AN APPLICATION HOSTING COMPANY FOCUSED ON OPEN SOURCE APPLICATIONS MANAGEMENT IN HIGH AVAILABILITY ENVIRONMENT.

NET OPEN SERVICES IS PROUD TO PROVIDE A HIGH QUALITY SERVICE TO OUR CUSTOMERS SINCE 10 YEARS.

OUR EXPERTISE INCLUDES:

- **CLOUD COMPUTING, PUBLIC, PRIVATE AND HYBRID CLOUD MANAGEMENT (OPENSTACK, CLOUDSTACK, RED HAT ENTERPRISE VIRTUALIZATION)**
- **REMOTE MONITORING AND MANAGEMENT 24/7**
- **NETWORKING AND SECURITY (OPEN BSD, IP TABLE, CHECKPOINT, CISCO,...)**
- **OS AND APPLICATION MANAGEMENT (FREE BSD, OPEN BSD, SOLARIS, UNIX, LINUX, AIX, MS WINDOWS)**
- **DATABASE MANAGEMENT (ORACLE, MYSQL, CASSANDRA, NOSQL, MS SQL, SYBASE...)**
- **MANAGED HOSTING IN CARRIER CLASS DATA CENTERS**
- **DISASTER RECOVERY**

Net Open SERVICES

WE PROVIDE SERVICES IN EVERY STEP OF THE PROJECT LIFE, DESIGN, DEPLOYMENT, MANAGEMENT AND EVOLUTIONS. **NETOPENSERVICES** TEAM INCLUDES EXPERIENCED LEADERS AND ENGINEERS IN THE INTERNET SERVER INDUSTRY.

OUR TEAM HAS **15 YEARS OF EXPERIENCE** IN DEVELOPING INTERNET INFRASTRUCTURE-GRADE SOLUTIONS AND PROVISIONING INTERNET DATACENTERS AND GLOBAL SERVICE NETWORKS TOGETHER.

WE OFFER EXCEPTIONAL HARDWARE SUPPORT AS SOFTWARE SUPPORT ON UNIX/LINUX AND OPEN SOURCE APPLICATION. **NETOPENSERVICES** DELIVERS THESE CUSTOM-BUILT LINUX AND UNIX SERVERS, AS WELL AS PRECONFIGURED SERVERS AND SCALABLE STORAGE SOLUTIONS, TO OUR CUSTOMERS. WE ALSO OFFER CUSTOM DEVELOPMENT AND ADVANCED-LEVEL UNIX/LINUX CONSULTING SOLUTIONS.

WWW.NETOPENSERVICES.COM • CONTACT@NETOPENSERVICES.COM

Great Specials

On FreeBSD® & PC-BSD® Merchandise

Give us a call & ask about our
SOFTWARE BUNDLES
1.925.240.6652

$39.95
FreeBSD 9.1 Jewel Case CD Set
or FreeBSD 9.1 DVD

$29.95
PC-BSD 9.1 DVD

$49.95
Save a BUNDLE!
The PC-BSD 9.0 Users Handbook
PC-BSD 9.1 DVD

$99.95
The FreeBSD CD **or** DVD Bundle

Inside each CD/DVD Bundle, you'll find:
FreeBSD Handbook, 3rd Edition
Users Guide FreeBSD Handbook, 3rd Edition, Admin Guide
FreeBSD 9.1 CD or DVD set
FreeBSD Toolkit DVD

Stylish Dress Attire
Look Your Professional Best
Comfy Apparel
Stay Warm in Zip Ups & Pullovers
T-Shirts
Lots of Styles to Choose From

FreeBSD 9.1 Jewel Case CD/DVD $39.95

CD Set Contains:

Disc 1	Installation Boot LiveCD (i386)
Disc 2	Essential Packages Xorg (i386)
Disc 3	Essential Packages, GNOME2 (i386)
Disc 4	Essential Packages (i386)

FreeBSD 9.0 CD	$39.95
FreeBSD 9.0 DVD	$39.95

FreeBSD Subscriptions

Save time and $$$ by subscribing to regular updates of FreeBSD

FreeBSD Subscription, start with CD 9.1	$29.95
FreeBSD Subscription, start with DVD 9.1	$29.95
FreeBSD Subscription, start with CD 9.0	$29.95
FreeBSD Subscription, start with DVD 9.0	$29.95

PC-BSD 9.1 DVD (Isotope Edition)

PC-BSD 9.1 DVD	$29.95
PC-BSD Subscription	$19.95

The FreeBSD Handbook

The FreeBSD Handbook, Volume 1 (User Guide)	$39.95
The FreeBSD Handbook, Volume 2 (Admin Guide)	$39.95

The FreeBSD Handbook Specials

The FreeBSD Handbook, Volume 2 (Both Volumes)	$59.95
The FreeBSD Handbook, Both Volumes & FreeBSD 9.1	$79.95

PC-BSD 9.0 Users Handbook	$24.95
BSD Magazine	$11.99
The FreeBSD Toolkit DVD	$39.95
FreeBSD Mousepad	$10.00
FreeBSD & PCBSD Caps	$20.00
BSD Daemon Horns	$2.00

Bundle Specials!
Save $$$

Just Plain Fun
Mousepads & Novelty Horns

BSD Magazine
Available Monthly

FreeBSD Mall

For even MORE items
visit our website today!
www.FreeBSDMall.com

iXsystems®

Headquarters:
San Jose, CA

855.GREP.4.IX | Contact Us

99% Compatibility

online now...

IXSYSTEMS AND YOU ARE THE PERFECT MATCH

SHARED INTERESTS

- ✓ Enterprise Storage Solutions
- ✓ Personalized Customer Service
- ✓ Bold New Information Technology

I'm a	Storage Reseller
In	The EU
Looking for	Storage Solutions to Sell
	A Technology Partner
	More Technical Experience
	New Business Opportunities

Visit Today!

iXsystems

Technology Partner Seeking Resellers/Integrators for TrueNAS™ Storage Appliance

WWW.IXSYSTEMS.COM/PERFECTMATCH